Assessing the Resilience of Automotive Intrusion Detection Systems to Adversarial Manipulation

STEFANO LONGARI, PAOLO CERRACCHIO, MICHELE CARMINATI, and STEFANO ZANERO, Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy

The security of modern vehicles has become increasingly important, with the controller area network (CAN) bus serving as a critical communication backbone for various Electronic Control Units (ECUs). The absence of robust security measures in CAN, coupled with the increasing connectivity of vehicles, makes them susceptible to cyberattacks. While intrusion detection systems (IDSs) have been developed to counter such threats, they are not foolproof. Adversarial attacks, particularly evasion attacks, can manipulate inputs to bypass detection by IDSs. This paper extends our previous work by investigating the feasibility and impact of gradient-based adversarial attacks performed with different degrees of knowledge against automotive IDSs. We consider three scenarios: white-box (attacker with full system knowledge), grey-box (partial system knowledge), and – the more realistic – black-box (no knowledge of the IDS' internal workings or data). We evaluate the effectiveness of the proposed attacks against state-of-the-art IDSs on two publicly available datasets. Additionally, we study effect of the adversarial perturbation on the attack impact and evaluate real-time feasibility by precomputing evasive payloads for timed injection based on bus traffic. Our results demonstrate that, besides attacks being challenging due to the automotive domain constraints, their effectiveness is strongly dependent on the dataset quality, the target IDS, and the attacker's degree of knowledge.

CCS Concepts: • **Security and privacy → Intrusion detection systems**; • **Computer systems organization** → *Embedded and cyber-physical systems*; • **Computing methodologies → Machine learning**.

Additional Key Words and Phrases: Automotive Security, Intrusion Detection Systems, Evasion Attacks

ACM Reference Format:
Stefano Longari, Paolo Cerracchio, Michele Carminati, and Stefano Zanero. 2018. Assessing the Resilience of Automotive Intrusion Detection Systems to Adversarial Manipulation. *J. ACM* 37, 4, Article 111 (August 2018), 26 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The security of modern vehicles has become an increasingly critical concern as the integration of connected technologies, such as Bluetooth and 5G, exposes vehicles to potential cyber threats. The CAN bus serves as the communication backbone for various electronic control units (ECUs), managing essential vehicle functions ranging from engine control to infotainment. However, the CAN protocol lacks fundamental security features such as encryption and authentication, leaving it vulnerable to cyberattacks. This issue was prominently highlighted in 2015 by Miller and Valasek, who remotely attacked a Jeep Cherokee [25, 26]. IDSs have been widely adopted as a key defense

Authors' Contact Information: Stefano Longari, stefano.longari@polimi.it; Paolo Cerracchio, paolo.cerracchio@mail.polimi.it; Michele Carminati, michele.carminati@polimi.it; Stefano Zanero, stefano.zanero@polimi.it, Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, , Italy.

ACM 1557-735X/2018/8-ART111
https://doi.org/XXXXXXX.XXXXXXX

mechanism to monitor network traffic and identify malicious activities, mainly through frequency-based detectors, analyzing patterns like packet timing or sequence, and payload-based detectors, examining packet content for irregularities. Traditionally, IDSs rely on machine learning models to detect anomalies [23, 32], but these models are susceptible to adversarial attacks, wherein carefully crafted inputs evade detection [38]. Current research on adversarial threats, however, mainly focuses on computer vision, with relatively few specialized studies in the automotive sector [2, 22].

In our previous work [2], we explored the resilience of automotive IDSs against white-box and grey-box adversarial attacks, where the attacker has either full or partial knowledge of the target system and data. The evasion attacks leverage gradient-based techniques, algorithms that were inspired from the computer vision domain but adapted to the automotive one, to craft adversarial examples that bypass IDS detection. However, in a real-world setting, attackers often lack direct access to the internal workings of the IDS, making black-box attacks a more realistic threat. Black-box attacks occur when the adversary has no knowledge of the IDS's architecture, parameters, or training data, and must rely either on probing the original system or on a surrogate model to infer attack strategies. Thanks to the transferability property of gradient-based Techniques [39], we focus on the ability of adversarial examples generated by surrogate models to transfer to grey- and black-box target IDSs. Given this premise, this paper extends our prior work by investigating the feasibility and impact of white-, grey-, and black-box adversarial attacks on automotive IDSs.

Our experimental evaluation on the ReCAN [47] and CARHacking [35] datasets provides a comprehensive evaluation of adversarial robustness of automotive IDSs. Our objective is to identify the effectiveness, impact, and feasibility of adversarial attacks in the automotive domain. Through our experiments, we first explore the vulnerability of IDS models in white- grey- and black-box scenarios, identifying the capability of adversarial algorithms to evade them. Then, we study the impact of adversarial perturbations on the network signals, to evaluate whether the meaning of the attack is maintained after the perturbation. Finally, we evaluate the feasibility of executing such attacks in real-time constrained systems by precomputing evasive attack sequences and identifying injection points in the CAN data stream. Our results demonstrate that the attack's effectiveness is strongly dependent on the dataset quality, the target IDS, and the attacker's degree of knowledge. As expected, white-box attacks proved overall most effective, degrading IDS performance by up to 60% on the most challenging dataset. Nonetheless, the impact of adversarial samples in grey- and black-box scenarios achieve lower but impactful results of up to 39% and 43% respectively. Interestingly, in the grey- and black-box scenarios, perturbation sometimes led to an increased detection rate, suggesting that in such constrained knowledge scenarios, the attempted perturbations may make the original attack more conspicuous to the target IDS. These aspects suggests that knowledge of the target IDS and the quality of data are crucial. The various models perform very differently, with autoencoder-based IDSs being more resistant to adversarial attacks, but predictor-based oracles being the more effective at generating adversarial samples. We conclude by discussing mitigations and the challenges to face in order to apply them. In summary, our contributions are the following:

- We investigate the feasibility and impact of adversarial attacks on payload-based automotive IDSs and different datasets, considering an attacker with different degrees of knowledge.
- We explore the transferability of adversarial examples generated by substitute models to grey- and black-box target IDS, addressing a more realistic and challenging scenario.
- We evaluate the impact of adversarial perturbations on the actual vehicle signals, providing a qualitative assessment of the attack's effectiveness in the different scenarios.

2 Background on CAN Security

CAN Primer. The controller area network (CAN) is a multi-master, message-broadcast protocol originally developed by Robert Bosch in 1986, and later updated and standardized in 2012 (ISO 11898-1). It is one of the most widely adopted network protocols and has become the de-facto standard for vehicle onboard networks, primarily used in the automotive industry to connect various ECUs within a vehicle. The CAN bus [6] operates as a broadcast medium, offering multi-master capabilities. This allows any node connected to the bus to read any packet transmitted across the network. Additionally, when the bus is idle, any node can transmit a message. If multiple nodes attempt to communicate simultaneously, message arbitration is resolved by prioritizing the message with the lowest ID. The CAN protocol incorporates various error checking mechanisms. If a message incurrs in an error it generates an error frame and invalidates the packet. Should errors persist, the malfunctioning node eventually removes itself from the network [15].

Four types of messages can be transmitted on a CAN network: *Data*, *Remote*, *Error*, and *Overload* frames. The *Data* frame is the most prevalent and, as its name implies, is used to transmit payload data. *Remote* and *Overload* frames are less frequently encountered in modern CAN systems.

Error frames are sent by either the transmitter or the receiver when an error is detected in the packet being transmitted on the bus, signaling that the current packet is invalid, and are not usually notified by the CAN controller to higher layer computation units.

Data frames payloads embed multiple values each, each with a distinct meaning. In this paper, we refer to these values as signals. Signals typically convey information from sensors or commands for actuators, but they can also contain noisy bit sequences or cyclic redundancy checks (CRCs), complicating their interpretation for security researchers. The mapping of these signals to specific CAN IDs is often proprietary information of the manufacturers, which do not disclose them. This reliance on security through obscurity presents significant challenges in the design of IDSs for CAN.

CAN Security Issues. CAN, originally designed with network isolation in mind, lacks any inherent security mechanisms and is thus vulnerable to various security threats [1]. Lacking *authentication* and *encryption*, and being a broadcast network, any node can intercept messages and perform actions aimed at compromising the authenticity and integrity of the transmitted data. Such as transmit spoofed messages with manipulated CAN IDs. Moreover, due to CAN's arbitration system, which prioritizes messages based on CAN IDs, an attacker can *misuse the protocol* by sending messages with low CAN IDs (higher priority), thus preventing legitimate messages from winning arbitration and causing a denial of service (DoS) attack. Recent approaches to CAN attacks focus triggering the error-handling mechanisms of CAN to silence specific nodes. This results in a targeted DoS attack where only the victim node is excluded from the bus [7].

ML-based Intrusion Detection in CAN. Intrusion detection is a widely studied field [8]. For a comprehensive review of automotive IDSs, refer to [19]. This section provides a concise overview of key concepts. Most CAN intrusion detection systems focus on anomaly detection, which identifies deviations from normal behavior. This classification can effectively differentiate between legitimate and injected messages in a network. deep learning (DL) techniques often use autoencoders [4] or predictive models [42]. Autoencoders learn a compressed representation of input data, then attempt to reconstruct it, while predictive models forecast the next sequence element based on previous samples. Both approaches operate in an unsupervised framework, assuming deviations from the training data represent anomalies. The difference between the reconstructed or predicted sample and the actual target, known as the *anomaly score*, quantifies deviation from the learned distribution. A threshold is used to classify instances as anomalous or not, typically derived from errors in a dedicated non-anomalous thresholding set.

CAN IDSs can be flow-based, payload-based, or hybrid [29]. Flow-based detection exploits deterministic ECU packet patterns. Taylor et al.[41] show that an support vector machine (SVM) can detect malicious outliers using metrics such as frame count, average inter-arrival time, and Hamming distance. However, frequency-based methods struggle with impersonation attacks that mimic normal ECU behavior while manipulating data fields. Time-based models also yield comparable results[27, 43]. Flow-based convolutional neural networks (CNNs) detectors exist but require matrix-structured inputs, increasing complexity and vulnerability to adversarial attacks [14, 19, 37].

Payload-based IDSs address these limitations by detecting subtle impersonation attacks but are generally more complex. Early works [5, 16, 42] use 64-bit data fields as input for machine learning (ML) models. Taylor et al.[42] propose a predictive long short-term memory (LSTM) that performs well on synthetic attacks, while Tanksale[40] focuses on single signals in the data flow, though identifying fields like revolutions per minute (RPM) and brake position remains difficult due to proprietary obfuscation. CANet [13] employs an LSTM-based autoencoder with a separate interface for each CAN ID, concatenating outputs for final reconstruction via an feed-forward neural network (FFNN), leading to a complex model. CANnolo and CANdito [21, 23] simplify this by reconstructing payload windows for individual IDs, expanding on the reverse engineering of automotive data frames (READ) method [24].

3 Related Work

In this section, we focus on relevant works in the field of adversarial machine learning.

Adversarial machine learning is a branch of ML that focuses on studying attacks against ML algorithms. Goodfellow et al.[38] demonstrated that small perturbations applied to the input of a deep learning model can induce classification errors in many realistic settings. These specially crafted inputs are known as *adversarial examples*[10]. Such techniques can be exploited by malicious attackers in various ways, each with different objectives [3]: **(a)** Exploratory attacks involve probing a model that behaves as a black box to extract knowledge based on its responses to different inputs. **(b)** Evasion attacks are the most common and were the first to be studied; here, the adversary manipulates inputs to cause misclassification. **(c)** Poisoning attacks involve contaminating the training data, which in many real-world scenarios requires evading checks by human experts or by earlier versions of the targeted system, leading to mislabeling when such data is used in training.

This research field, originally stemming from computer vision, has also been applied to security-critical applications, such as network or transaction monitoring and malware detection [36]. Previous research has revealed interesting properties of adversarial examples: Papernot et al.[30] demonstrated the transferability of adversarial attacks by training an *oracle*—a substitute network—to carry out attacks instead of directly targeting the model. Longari et al.[22] designed a similar oracle-based approach within the domain of automotive IDSs, developing a greedy algorithm for black-box adversarial attacks. However, this work evaluates the distortion introduced in the perturbed packets using Hamming distance, which does not fully capture the actual semantic distance and overlooks attackers with varying levels of knowledge about the target system.

One immediate way to enhance the resilience of DL models in supervised or semi-supervised settings is adversarial training, which involves including labeled adversarial examples in the training set [45]. Another approach is to select more resilient input features; for example, Papernot et al. [31] propose training an initial model and then approximating it with a second, more resilient model, leveraging the confidence scores and class similarity insights from the first model.

In the context of network intrusion detection, Li et al.[20] attacked an in-vehicle Ethernet monitored by an LSTM IDS classifier[17] using fast gradient sign method (FGSM) and basic iterative method (BIM), achieving a recall score as low as 2%. The authors then retrained the LSTM with adversarial examples, nearly restoring the baseline attack-free score ($\sim$ 98%). Similarly, Sauka et

Table 1. Overview of attacker knowledge and capabilities across threat scenarios

	White-box	Grey-box	Black-box
Access to IDS model architecture	Full	None	None
Access to IDS training data	Full	Full	None
Oracle	Not required	Required	Required
Training dataset for oracle	-	Same as IDS	Similar but independent

al. [34] conducted multiple tests against FGSM, projected gradient descent (PGD), and successfully mitigated the attack using adversarial training.

4 Threat Model

As highlighted in prior research [19, 29], the specific objectives of attacks on the CAN bus can vary, but they generally involve injecting sequences of packets to achieve outcomes ranging from impairing vehicle functionality to compromising the safety of passengers and nearby individuals. These attacks typically work by introducing false sensor data or forged control commands. Given the lack of inherent security mechanisms in the CAN protocol, as discussed in Section 2, we assume an attacker with control over a node in the network, which is plausible given the established methods for carrying out complex attacks on CAN systems. This level of control allows the attacker to inject or remove packets to obtain its goals.

We define an **evasive attack** as one in which the injected packets maintain their malicious intent (e.g., spoofing sensor values or control commands) while being purposefully crafted to avoid detection by the IDS. In other words, our adversarial strategy focuses on perturbing a predetermined set of malicious packet sequences with the goal of transforming them into evasive examples. The attacker then - controlling a node on the bus - injects the sequence of perturbed packets on the bus. To perturb them effectively, an attacker may have black-, grey-, or white-box knowledge of the target IDS, each with varying degrees of access and requirements. In Table 1 we provide a summary of the core properties of each attacker, of which a more detailed explanation follows:

Black-box Attacker. The attacker has no access to the internal architecture or training data of the target IDS. To generate adversarial inputs, they rely on a surrogate (oracle) model [30] trained to approximate the target IDS's behavior. This surrogate is trained on a separate but compatible dataset, which could realistically be collected from a similar vehicle—e.g., one of the same make and model. The generated adversarial examples are then transferred with the goal of evading detection by the target IDS. In our experiments, this scenario is simulated by splitting the dataset into two disjoint subsets: one used to train the surrogate model and the other for the target IDS.

Grey-box Attacker. In the grey-box scenario, the attacker has access to the training dataset used by the target IDS, but no knowledge of its internal architecture or parameters. This access could be obtained through a data leak or breach, though the specific method is beyond the scope of this work. The attacker uses this dataset to train a surrogate model that approximates the target's decision boundary, and generates adversarial samples through the surrogate model.

White-box Attacker. The attacker has full knowledge of the target IDS, including its model architecture, parameters, training data, and expected behavior of the monitored node. This privileged access allows them to accurately predict traffic from the victim ECU and craft coherent, stealthy attack sequences that blend into the observed communication. Such a scenario typically assumes insider access or deep system compromise.

5 Motivation

In the automotive domain, machine learning models are already being used and have been proposed for tasks that directly affect vehicle safety, efficiency, and security. Ensuring the security of these

models is therefore critically important. Within this context, Adversarial Machine Learning (AML) attacks pose a unique threat due to their ability to algorithmically generate evasive examples designed specifically to bypass detection mechanisms.

To provide a complete assessment of evasion feasibility in automotive domains, unlike previous studies where adversarial samples are evaluated in ideal scenarios, this work emphasizes creation of samples coherent with the concurrent non-anomalous network traffic. Additionally, we aim to preserve characteristics typical of automotive attacks, which often do not involve a single packet injection but rather sequences of harmful packets, while attempting to minimize deviation from the original attack sequence. Specifically, we evaluate the effectiveness, feasibility, and impact of these attacks in the automotive context by addressing three key aspects: (i) their ability to evade state-of-the-art intrusion detection systems, (ii) their capacity to preserve the intended malicious semantics, and (iii) the possibility of overcoming the real-time constraints of the CAN bus through the precomputation of evasive attack sequences.

Our previous work. The foundation for this research is laid by our previous work, Cerracchio et al. [2], which began investigating the impact of evasion attacks on automotive IDSs. We adapted well-known gradient-based adversarial techniques from the computer vision domain to the automotive context, focusing on payload-based IDSs that leverage machine learning models. Our findings highlighted the feasibility of such attacks and emphasized the role of model complexity and attack quality in determining the success of evasion attempts. Building on these results, the present work extends the investigation by exploring three attacker scenarios, white-box, grey-box, and black-box, integrating the latter, which is the most realistic attack setting, where adversarial samples are crafted using a surrogate detection model trained on data from a similar vehicle, and evaluated on the target IDS. To simulate varying degrees of attacker knowledge, we re-executed all experiments using a reduced dataset[1], to provide consistent results across the various attacker models. Additionally, we replicated the experiments on a second dataset to assess how dataset variability influences adversarial behavior. The combination of an additional dataset and a systematic exploration of all three attacker scenarios allows us to draw clearer conclusions about the feasibility of adversarial attacks on CAN-based IDSs. Finally, we integrate the work with a discussion on countermeasures.

6 Approach

The objective of our study is to investigate how evasion attacks can be adapted to the domain of automotive intrusion detection and to assess their effectiveness. In doing so, we aim to identify attack strategies that are compatible with the unique characteristics of this domain — namely, the tabular and temporal nature of the data, the sensor-like behavior of certain signals within the payloads, and the real-time constraints imposed by the CAN bus environment. Our approach consists of two key steps: first, we perform a domain-specific preprocessing phase that transforms raw CAN payloads into a representation suitable for machine learning-based IDSs. Second, we adapt and extend state-of-the-art gradient-based adversarial techniques—originally developed for computer vision—to align with the constraints of automotive systems. These adaptations include limiting perturbations to physically plausible ranges, preserving signal semantics, and lowering the average convergence iterations to increase compatibility with real-time execution. We opt for gradient-based approaches due to their transferability property [39]: being it unlikely for an attacker to possess the actual models running on the target IDS, the capability of gradient-based attacks to transfer to other models comes helpful to allow for the training of the attacks against an oracle and their execution against a different target model.

[1]The original datasets were split into two sets each, simulating scenarios with different levels of attacker knowledge.

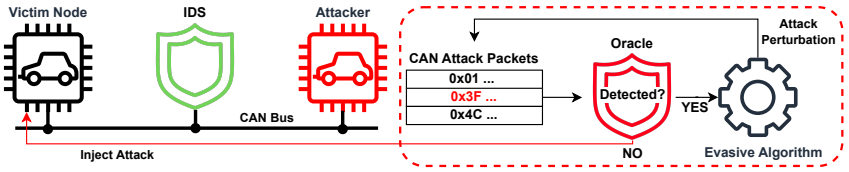


Fig. 1. Scheme of the attacker's approach. The attacker controls a node with bus access. Either in real-time or precomputing it, they generate the evasive attack sequence through the oracle and inject it on the CAN bus.

Table 2. Bit ranges with semantic meaning identified

Type	Description
Constant	Bits that remain constant. They are excluded from the feature vector.
Physval	Sequences of bits that contain a changing value, usually corresponding to physical signals.
Binary	Ranges containing an isolated, non-constant bit, interpreted as a logical flag.
CRC	Checksums for message integrity, detected through their Gaussian random distribution.
Counter	Application-level counters, increasing by one with each subsequent frame.

Preprocessing. To ensure domain-relevant input representations, we perform a preprocessing step that extracts semantically significant features from raw payloads, addressing the lack of publicly available signal mappings and the obfuscated nature of CAN data. Following the approach proposed in prior works [23, 47], we apply a heuristic analysis to the packet payloads, incorporating slight improvements over the original READ method [24]. This analysis enables the identification of bit ranges with distinct semantic roles by examining how frequently bits change over time. For example, it distinguishes adjacent bits that function as counters from isolated binary flags that do not exhibit strong correlation with neighboring bits. Using attack-free data, we categorize the extracted fields into five types, as summarized in Table 2. Consistent with findings from previous work [21], we retain only physical and binary ranges as features, which improves both reconstruction accuracy and model efficiency compared to using the full set of non-constant bits. Finally, physical signals are parsed as integers and normalized in the $[0, 1]$ interval by dividing each value by the maximum representable number for its corresponding bit length.

Evasion Algorithms. In our approach [2], as visible in Figure 1, the attacker starts with an initial CAN log containing some unperturbed attack messages as a baseline, and attepmts to morph these sequences to be evasive by applying repeated modifications. We formalize an evasion attack as the optimization problem of finding the adversarial sample $\tilde{x}$, undetected under the discrimination function $F(x)$ while minimizing the perturbation $\delta(\tilde{x}, x)$: $\tilde{x} = argmin_{x^*}[\delta(x^*, x)]$ s.t. $F(x^*) = 0 \wedge x^* \in D_x$. In most applications, the generated adversarial sample $\tilde{x}$ must also comply with domain-specific constraints, i.e., $\tilde{x} \in D_x$. For instance, in the computer vision domain, inputs are typically constrained to pixel intensity values within the integer range $[0, 255]$. In our case, we enforce domain validity by restricting each perturbed, normalized signal obtained during the preprocessing stage to remain within the $[0, 1]$ interval.

The algorithms we propose are derived from gradient-based techniques originally designed for computer vision; the common rationale behind these kinds of techniques, namely fast gradient method (FGM), BIM [18], and Deepfool [28], is to leverage the backpropagation algorithm against the model under attack. The intuition is to push the original input towards areas in the problem space with lower confidence, approaching and crossing the decision boundary by exploiting the gradient ascent of an arbitrary loss function. Figure 1 illustrates a single iteration of the process. At each iteration, the chosen algorithms try to find an optimal additive term and produce x^{t+1}.

6.1 BIM-based algorithms

Algorithm 1: Proposed BIM step decay variant

```
begin
    t ← 0;
    step ← ϵ;
    while t < max_iter do
        score ← get_score(ids_model, sample);
        grad ← gradient_sample(score);
        pert ← step · sign(grad) ;                                  // (1)
        pert ← −pert · tamper_mask ;                                // minimizing
        sample ← clip_min,max(sample + pert);
        sample ← round(sample);
        step ← step · decay ;                                       // step decay
        if get_score(ids_model, sample) < threshold then
            return sample ;                                         // sample is now evasive
        end
        t ← t + 1;
    end
    return None ;                                                   // Abort computation
end
```

Algorithm 2: Proposed $l2$ BIM variant

```
begin
    t ← 0;
    step ← ϵ;
    while t < max_iter do
        score ← get_score(ids_model, sample);
        grad ← gradient_sample(score);
        pert ← step · grad/||grad||_2 ;                             // (2)
        pert ← −pert · tamper_mask ;                                // minimizing
        sample ← clip_min,max(sample + pert);
        sample ← round(sample);
        if get_score(ids_model, sample) < threshold then
            return sample ;                                         // sample is now evasive
        end
        t ← t + 1;
    end
    return None ;                                                   // Abort computation
end
```

The original BIM iteratively applies the FGSM perturbation described by (1).

$$x^{t+1} = clip_X(x^t - \epsilon \frac{\nabla_x L(w, x^t)}{||\nabla_x L(w, x^t)||}) \tag{1}$$

From a geometric point of view, this method produces a perturbation directed towards a maximum of the loss function L via approximation of the gradient $\nabla_x L$ and constrained by $||x^{t+1} - x^t||_\infty \leq \epsilon$, depending on the hyperparameter ϵ.

Algorithm 3: Pseudocode for the DeepFool variant

```
begin
    t ← 0;
    step ← ϵ;
    while t < max_iter do
        score ← get_score(ids_model, sample) − threshold;
        grad ← gradient_sample(score);
        pert ← (score·grad) / ||grad||_2^2 ;                                  // (3)
        pert ← (1 + ϵ)pert · tamper_mask ;                                    // projection
        sample ← clip_min,max(sample + pert);
        sample ← round(sample);
        if get_score(ids_model, sample) < 0 then
            return sample ;                                                   // sample is now evasive
        end
        t ← t + 1;
    end
end return None ;                                                             // Abort computation
```

We implement two slightly different variants of the BIM attack, namely the *step decay* BIM and the $l2$ BIM; both versions of the algorithm include an adjustment procedure to restrict the resulting value x^{t+1} to stay in the problem space. We achieve this by clipping and rounding to the nearest integer so that the features can be effectively represented in the actual underlying bit vector. This ensures that modified payloads remain within valid signal representations, aligning with the bit-level constraints and signal encoding formats typical of in-vehicle CAN traffic. Besides this modification, the *step decay* method differs from the simple BIM in the parameter ϵ: in our implementation, it has not a fixed magnitude but rather a geometrically decreasing value according to the update rule $\epsilon^{t+1} = \epsilon^t * \omega$, introducing the hyperparameters ϵ_0 and ω. The reason for the introduction of *step decay* is that, similarly to what happens with the learning rate decay during the training of DL models [46], the introduction of a geometrically decreasing step size helps mitigate oscillations around the decision boundary, reducing the number of iterations needed and making the attack more compatible with the strict real-time constraints of automotive systems.

$$x^{t+1} = clip_X(x^t - \epsilon \frac{\nabla_x L(w, x^t)}{||\nabla_x L(w, x^t)||_2}) \tag{2}$$

Conversely, the $l2$ BIM simply uses the Euclidean norm instead of the absolute value for the FGSM equation, resulting in (2). In this case, ϵ quantifies the module of the perturbation vector, which now completely orients itself according to the gradient and constitutes the tighter bound $||x^{t+1} - x^t||_2 \leq \epsilon$. We deem this approach more suitable in our case - given the higher dimensionality and feature inter-correlation in the automotive domain - but also in general when dealing with diverse features, as they apply small perturbations along all dimensions. By relying on the Euclidean norm, the l_2 BIM distributes the perturbation more evenly across all features—an approach particularly well-suited for the highly interdependent and heterogeneous signals found in automotive payloads. In both implementations, we choose as loss function the opposite of the anomaly score.

6.2 DeepFool-based algorithm

Deepfool [28] is another iterative method that approximates the IDS as an affine classifier $w \cdot x + b = F(w, x)$, then the perturbation δ tries to push the sample beyond the affine decision boundary,

which we assume to be the 0 plane for F.

$$x^{t+1} = x^t + (1+\epsilon) \cdot (-F(w,x)\frac{\nabla_x F(w,x^t)}{||\nabla_x F(w,x^t)||_2^2})) \tag{3}$$

Equation (3) illustrates the original computation, notice the overshooting factor $1+\epsilon$: since the algorithm can't converge to a point precisely on the decision boundary, we attempt to cross it by a small value ϵ instead - according to the approximation - and cause the objective misclassification. In our case, we do not deal with a sign-dependent decision value like $F(w,x)$, however, we can consider the reconstruction error $L(w,x)$ as a classification confidence score and apply a shift by the thresholding value θ to obtain an analogous zero-centered boundary $F(w,x) = L(w,x) - \theta$. We apply the same adjustment procedure to obtain valid samples that we use for the BIM algorithms. This may hinder the algorithm convergence; however, we mitigate this phenomenon by testing different overshooting magnitudes. According to the evaluation of the original paper, albeit more computationally complex, it should terminate in fewer iterations than simple BIM. Note that in most automotive intrusion detection algorithms, the input windows contain malicious and legitimate packets. This is true for all the models in our experimental evaluation except the FFNN. Therefore, we apply the computed perturbation only on the injected packages at each step via a simple projection of the computed perturbation matrix. Notably, in this way the algorithm naturally aligns with the temporal nature of predictive IDSs: the attacker perturbs the most recent packet to better match the model's prediction, and upon successful evasion, the sliding input window incorporates this crafted packet, subsequently affecting future classifications.

7 Experimental Evaluation

In this section, we describe the experiments and discuss the results that are the core of our investigation, which aim to answer the following research question:

Are existing adversarial evasion attacks effective against payload-based automotive IDSs?

The *first experiment* evaluates the feasibility and impact of adversarial attacks against IDSs performed by an attacker with different degrees of knowledge (see Section 4) of the target system, using the ReCAN [47] and CarHacking [35] datasets.

The *second experiment* evaluates whether the adversarially perturbed attacks maintain the attack goal, assessing the effectiveness of the adversarially perturbed attacks on the vehicle by comparing the shape of the original and adversarial attacks signals.

Finally, the *third experiment* is meant to study whether attacks can be executed even in stringent real time constraints, by restricting the assumption of the attacker's computational capabilities. To emulate the physical real-time constraints, we precompute the evasive payload sequence allowing an attacker to inject it entirely when given requirements on bus traffic are met.

7.1 Experimental Settings

Performance Metrics. Since the goal of this work is to assess the impact of adversarial evasion attacks across different IDS architectures, the attacker does not alter benign traffic. As a result, the IDSs' behavior on legitimate (non-malicious) traffic remains unchanged. Consequently, metrics such as true negatives (TN) and false positives (FP) are unaffected by the attack and are therefore not relevant to our evaluation. We mainly report the true positive rate (TPR) (i.e., recall) of each IDS under attack, as it directly reflects the degradation in detection capability due to adversarial perturbations. Moreover, to quantify the magnitude of the adversarial perturbation applied to evade detection, we introduce the *aggregate perturbation (AP) metric*, which measures the *mean maximum*

perturbation of a single field in each packet. Formally, we define it as:

$$\mathsf{AP} = \frac{\sum_{i=1}^{N} ||x_i - \tilde{x}_i||_\infty}{N}$$

where N is the number of malicious packets in the test set, x_i is the array of features in the original i-th malicious packet, and $\tilde{x}_i$ is the corresponding adversarial packet. The infinity norm captures the largest individual feature perturbation per packet. Since all features are normalized to the $[0, 1]$ range, the AP score also lies within this interval.

7.2 Datasets Under Analysis

For our evaluation we rely on two publicly available, real-world traffic datasets: ReCAN [47] and the Car Hacking Dataset (CH) [35]. Both datasets offer realism in the untampered section of the data, being extracted live from real vehicles. Where the ReCAN dataset integrates synthetic attacks to allow for complex attack events to be tested, the CH dataset contains real-world attacks, which are however executed live with safety in mind and therefore less complex and predictable.

7.2.1 ReCAN Dataset. The *ReCAN dataset* [47] consists of real CAN traffic collected from a Giulia Veloce vehicle using a CANtact interface [9]. We focus on the "C-1" subset, which provides the richest trace (over two hours of city and highway driving). This dataset contains no attack instances, enabling us to inject *synthetic* attacks in a controlled and isolated manner, in line with prior work [21, 23].

Synthetic attacks are generated using the *CANtack* tool and following the same setup presented by Nichelini et al. [29], which supports both injection and masquerade strategies. Specifically, the following attack types are included:

Injection Replay: Injects previously observed packets at a lower rate (injection rate = 0.4) to avoid triggering frequency-based detection.

Masquerade Replay: Simulates ECU impersonation, replaces 25 valid packets content with content of a previously recorded sequence of valid packets, without affecting frequency or period of arrival.

Continuous Change: Replaces 25 legitimate packets content without affecting frequency or period of arrival, but instead of replaying previous data it gradually modifies a single signal (of at least 9 bits) over 25 packets to reach a random target value inside the boundaries of the signal.

Change to Minimum: A variant of the above, targeting a final all-zero value.

Fuzzy: Randomly modifies physical and binary fields at each packet.

In the context of CAN security, injection attacks typically involve the insertion of new packets by the attacker, superimposed on the existing traffic. This often results in a noticeable change in the arrival frequency of the attacked packets, which could be detected. In contrast, masquerade attacks usually refer to silencing the sender ECU by various methods [7, 26], and then sending packets on its behalf at the correct frequency to avoid detection by frequency-based IDSs. To further hide their presence, attackers may execute attacks that do not abruptly change signal values but that do so gradually. Finally, fuzzy attacks are usually meant to test a system for unexpected behavior, modify the values of the payload randomly, and are usually relatively simple to detect. All generated attacks are signal-aware and generated using an enhanced version of the READ heuristic [24], modifying only the targeted field within each packet. Each attack is independently generated per CAN ID and spaced at least one minute apart. Following CANova [29], we select 12 specific IDs for evaluation. To prevent overfitting, training and testing sequences are drawn from disjoint portions of the dataset. In black-box settings, oracles and IDSs are trained on separate sequences to simulate differing attacker and defender environments.

7.2.2 Car Hacking Dataset. The publicly available car hacking (CH) dataset [35] complements ReCAN by offering real-world adversarial scenarios collected via the on-board diagnostics (OBD) port during live vehicle operation. It includes five experiments: DoS, fuzzy, RPM spoofing, gear spoofing, and an attack-free trace. We focus on the RPM and gear spoofing attacks (IDs `0316` and `043f`) as they involve *payload-level manipulations* and are conducted in a realistic setting.

Unlike synthetic scenarios, these attacks were executed live with safety in mind. As such, they are inherently less aggressive, need to provide predictable outcomes (e.g., only specific IDs are attacked) and must maintain vehicular safety during execution. Nevertheless, they provide valuable insights into how adversarial perturbations behave in practical contexts.

We exclude the DoS and fuzzy attacks from our evaluation, as they are trivially detectable [33, 35] and not aligned with the stealth-focused objective of our adversarial approach. Due to known inconsistencies between the "attack-free" and "injected" logs [44], we train all models using the untampered portions of the attack sessions themselves to avoid distribution shift.

7.3 Selected intrusion detection systems

Given that attacks that alter the frequency are easy to detect, an adversarial attacker will implement masquerade attacks that do not alter frequency of the network stream. Therefore, to assess the effectiveness of evasion attacks against IDSs in the automotive field, we select six commonly used [22] architectures of payload-based anomaly detector.

FFNN. A one-to-one autoencoder with two fully connected layers with 16 units each; this architecture is blind to replay attacks as it considers a single packet at a time, however, it is a simple solution to detect more obvious payload tampering and is included to provide a baseline reference.

CANdito [21]. A window-to-window symmetrical autoencoder with two fully connected (128 units each) and two LSTM layers (64 cells each), it is the most complex among the chosen networks.

LSTM predictors [42]. Two window-to-one predictors, we implement two variants, a short with just two LSTM layers having 32 cells each, and a long with four LSTM layers, with 64 and 16 cells.

GRU-based predictors. Two window-to-one predictors analogous to the short and long LSTM variants, employing the more lightweight gated recurrent units (GRUs) instead.

A final sigmoid-activated dense layer with one unit per input feature follows each individual architecture to provide an output with the correct dimensionality. While the predictor models produce an anomaly score for one packet at a time, with a rolling input window, the CANdito autoencoder reconstructs the whole window, operating with non-overlapping input sequences. We choose a window size of 40 (or 39 plus one predicted frame for the predictor models).

7.4 Experiment 1: Adversarial attack performance evaluation

7.4.1 Baselines Performance. Table 3 reports the average recall and precision for various non-evasive attacks evaluated across two datasets—ReCAN and CarHacking—using a range of IDS models: FFNN, CANdito, shortLSTM, longLSTM, shortGRU, and longGRU. These metrics provide a baseline understanding of each model's ability to detect attacks under non-evasive conditions.

On the ReCAN dataset, results align with expectations. The *Fuzzy* attack, which significantly disrupts data patterns, yields high recall across all models, with CANdito showing the best performance. However, precision drops notably—especially in shortLSTM—indicating non null false positive rates (note that this is not particularly relevant for our experiments, since the adversarial samples do not affect non-tampered data). Attacks such as *Continuous Change* and *Change to Minimum* show moderate recall, particularly in CANdito and longLSTM, but again suffer from low precision, especially in shortLSTM and shortGRU. The *Injection Replay* and *Masquerade Replay* attacks are particularly challenging to detect, showing very low recall and precision across all models. For

Table 3. Average Recall and Precision for each non-evasive attack on the ReCAN and CarHacking Dataset.

	Attack	Metrics	FFNN	CANdito	shortLSTM	longLSTM	shortGRU	longGRU
ReCAN	Fuzzy	recall	0.9192	0.9777	0.8928	0.8430	0.8976	0.9058
		precision	0.7032	0.7758	0.4542	0.4264	0.5241	0.4763
	Continuous Change	recall	0.4317	0.7758	0.6321	0.6256	0.6224	0.6190
		precision	0.5424	0.7270	0.2874	0.3294	0.3478	0.3125
	Change to Minimum	recall	0.5158	0.7845	0.5307	0.5550	0.5224	0.5116
		precision	0.7152	0.9184	0.2852	0.2916	0.3761	0.2839
	Injection Replay	recall	0.0000	0.4215	0.6640	0.5447	0.6927	0.6893
		precision	0.0000	0.6513	0.2436	0.3268	0.2718	0.2494
	Masquerade Replay	recall	0.0006	0.3604	0.2743	0.2557	0.2070	0.2214
		precision	0.0067	0.5024	0.1522	0.1728	0.1690	0.1366
CarHack.	RPM Spoofing	recall	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
		precision	1.0000	1.0000	0.9066	0.9066	0.9066	0.9066
	Gear Spoofing	recall	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
		precision	0.9950	0.9655	0.9090	0.9090	0.9090	0.9090

instance, FFNN fails almost entirely to detect Injection Replay, due to its lack of temporal context, while sliding-window models perform slightly better, benefiting from sequential input awareness.

In contrast, the CarHacking dataset yields significantly stronger results. Both the *RPM Spoofing* and *Gear Spoofing* attacks are consistently detected, with all models achieving perfect recall. Precision is similarly high across the board, with CANdito slightly outperforming others on the *Gear Spoofing* attack. These results highlight that real-world injected attacks, potentially due to the safety limitations in generating the dataset, are more easily detected than synthetic ones, and the similarity in performance across the RPM and Gear spoofing attacks is expected, as both rely on the same injection-based attack strategy.

7.4.2 White-Box scenario. In this experiment, we compare the performances of all six IDSs over all the available attacks in the white-box scenario. Tables, 4, 5, 6, 7, and 8 show the aggregated results for the adversarial attacks on the ReCAN dataset, while Tables 9 and 10 present the results on the CarHacking dataset. Note that the performance metric values in each table are averaged over all evaluated IDs. Since we test the evasive packets generated by each oracle model against all the other available architectures on the same dataset, the diagonal in these tables represents the results of the white-box scenario, while the remaining part shows the results of the grey-box one.

ReCAN. Overall, the performances on the ReCAN datasets in the white-box scenario demonstrate the effectiveness of the evasion process. Across all the attacks, multiple algorithms manage to lower the detection rate of the respective detection systems by 10% to 60%. As presented in Table 11, the oracles in the white-box scenario significantly outperform the others in lowering the detection rate.

Among all attacks, the fuzzy attack (see Table 4) is predictably the easiest to detect due to the random fluctuations in the values of the tampered signals. The intent behind the fuzzing attack is often to investigate the behavior of a system or study its interaction with the IDSs, rather than to produce a specific effect. Notably, it is also the attack with the highest white-box evasion performance, with DeepFool repeatedly causing a detection rate loss exceeding 50% compared to the baseline. Interestingly, CANdito maintains a significantly lower performance loss than the other algorithms.

The continuous change and change-to-minimum attacks (see Tables 5 and 6) exhibit comparable baseline performances, with the former being slightly easier to detect (with a difference of about 5-10% in TPR), likely due to the randomly generated target value potentially falling outside the normal behavior of the signal. As expected, for similar reasons, it is also easier to find an evasive sample for the continuous change attack. In these attacks, the intruder replays previous packets

Table 4. Results on the ReCAN dataset in the white-box, grey-box (first part of the table, the white diagonal line represent the white-box), and black-box scenarios for the fuzzy attack.

White-box and Grey-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.02	0.81 (-11.21%)	0.98 (0.0%)	0.89 (-0.12%)	0.84 (0.07%)	0.9 (-0.06%)	0.9 (-0.29%)
	L2 BIM	0.18	0.43 (-48.99%)	0.98 (0.0%)	0.89 (-0.71%)	0.83 (-1.29%)	0.89 (-0.68%)	0.9 (-0.71%)
	DeepFool	0.19	0.58 (-33.67%)	0.98 (0.0%)	0.88 (-0.94%)	0.82 (-1.94%)	0.89 (-0.8%)	0.89 (-1.26%)
CANdito	Decay BIM	0.0	0.92 (0.0%)	0.98 (0.0%)	0.89 (0.0%)	0.84 (0.0%)	0.9 (0.0%)	0.91 (0.0%)
	L2 BIM	0.0	0.92 (-0.06%)	0.97 (-1.04%)	0.89 (0.0%)	0.84 (0.0%)	0.9 (0.0%)	0.91 (0.0%)
	DeepFool	0.01	0.91 (-1.06%)	0.89 (-9.03%)	0.89 (-0.17%)	0.84 (-0.23%)	0.9 (-0.17%)	0.9 (-0.17%)
Short LSTM	Decay BIM	0.0	0.92 (-0.0%)	0.98 (0.0%)	0.87 (-2.66%)	0.84 (-0.06%)	0.89 (-0.47%)	0.9 (-0.18%)
	L2 BIM	0.1	0.88 (-4.03%)	0.98 (0.0%)	0.66 (-22.82%)	0.81 (-2.88%)	0.86 (-3.48%)	0.87 (-3.14%)
	DeepFool	0.79	0.67 (-24.9%)	0.87 (-10.6%)	0.29 (-59.87%)	0.53 (-30.86%)	0.63 (-27.03%)	0.63 (-27.59%)
Long LSTM	Decay BIM	0.0	0.92 (0.19%)	0.98 (0.0%)	0.89 (-0.11%)	0.81 (-2.81%)	0.89 (-0.34%)	0.9 (-0.29%)
	L2 BIM	0.13	0.84 (-7.56%)	0.98 (0.0%)	0.87 (-2.77%)	0.58 (-26.48%)	0.86 (-3.33%)	0.87 (-3.61%)
	DeepFool	0.76	0.67 (-25.31%)	0.87 (-10.88%)	0.68 (-21.22%)	0.24 (-59.99%)	0.7 (-20.04%)	0.69 (-22.0%)
Short GRU	Decay BIM	0.0	0.92 (0.01%)	0.98 (0.0%)	0.89 (-0.12%)	0.84 (-0.18%)	0.88 (-2.16%)	0.91 (-0.07%)
	L2 BIM	0.1	0.9 (-2.12%)	0.98 (0.0%)	0.87 (-2.44%)	0.83 (-1.4%)	0.69 (-20.61%)	0.88 (-2.14%)
	DeepFool	0.7	0.74 (-17.93%)	0.9 (-8.04%)	0.67 (-22.21%)	0.64 (-20.73%)	0.35 (-54.59%)	0.7 (-20.63%)
Long GRU	Decay BIM	0.0	0.92 (0.07%)	0.98 (0.0%)	0.89 (-0.28%)	0.84 (-0.11%)	0.9 (-0.23%)	0.88 (-2.2%)
	L2 BIM	0.1	0.89 (-2.98%)	0.98 (0.0%)	0.85 (-3.87%)	0.82 (-2.62%)	0.86 (-3.77%)	0.69 (-21.16%)
	DeepFool	0.76	0.67 (-25.39%)	0.86 (-11.98%)	0.64 (-24.83%)	0.62 (-22.2%)	0.67 (-23.21%)	0.31 (-59.36%)
Black-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.02	0.9 (-2.12%)	0.98 (0.0%)	0.89 (0.0%)	0.84 (-0.04%)	0.9 (-0.07%)	0.9 (-0.11%)
	L2 BIM	0.12	0.85 (-7.18%)	0.98 (0.0%)	0.89 (-0.09%)	0.84 (-0.78%)	0.89 (-0.29%)	0.9 (-0.62%)
	DeepFool	0.16	0.84 (-7.6%)	0.98 (0.0%)	0.89 (-0.61%)	0.84 (-0.5%)	0.9 (0.0%)	0.9 (-0.36%)
CANdito	Decay BIM	0.0	0.92 (0.0%)	0.98 (0.0%)	0.89 (0.0%)	0.84 (0.0%)	0.9 (0.0%)	0.91 (0.0%)
	L2 BIM	0.0	0.92 (0.01%)	0.97 (-1.04%)	0.89 (0.0%)	0.84 (-0.0%)	0.9 (0.0%)	0.91 (0.0%)
	DeepFool	0.01	0.92 (-0.19%)	0.97 (-1.04%)	0.89 (-0.17%)	0.84 (0.06%)	0.9 (-0.17%)	0.9 (-0.17%)
Short LSTM	Decay BIM	0.0	0.92 (-0.13%)	0.98 (0.0%)	0.89 (-0.07%)	0.84 (0.01%)	0.9 (0.0%)	0.9 (-0.18%)
	L2 BIM	0.13	0.86 (-5.48%)	0.98 (0.0%)	0.87 (-2.33%)	0.82 (-2.33%)	0.87 (-2.36%)	0.88 (-2.62%)
	DeepFool	0.69	0.72 (-20.27%)	0.95 (-2.57%)	0.68 (-21.23%)	0.62 (-22.53%)	0.71 (-18.98%)	0.71 (-19.63%)
Long LSTM	Decay BIM	0.0	0.92 (0.0%)	0.98 (0.0%)	0.89 (0.01%)	0.84 (-0.1%)	0.9 (-0.06%)	0.91 (-0.06%)
	L2 BIM	0.15	0.85 (-7.36%)	0.98 (0.0%)	0.87 (-2.03%)	0.82 (-2.23%)	0.88 (-1.76%)	0.88 (-2.68%)
	DeepFool	0.74	0.65 (-26.82%)	0.94 (-4.24%)	0.65 (-24.48%)	0.59 (-25.4%)	0.7 (-19.48%)	0.71 (-19.43%)
Short GRU	Decay BIM	0.0	0.91 (-0.8%)	0.97 (-0.32%)	0.89 (0.12%)	0.83 (-0.93%)	0.9 (0.22%)	0.91 (-0.04%)
	L2 BIM	0.11	0.91 (-1.34%)	0.98 (0.0%)	0.88 (-1.36%)	0.83 (-1.24%)	0.87 (-2.32%)	0.89 (-1.92%)
	DeepFool	0.74	0.75 (-17.14%)	0.94 (-3.5%)	0.69 (-20.08%)	0.65 (-19.41%)	0.72 (-18.14%)	0.71 (-19.71%)
Long GRU	Decay BIM	0.0	0.92 (-0.11%)	0.98 (0.0%)	0.89 (-0.12%)	0.84 (0.01%)	0.9 (-0.17%)	0.91 (0.0%)
	L2 BIM	0.13	0.88 (-3.92%)	0.98 (0.0%)	0.87 (-1.96%)	0.82 (-2.07%)	0.87 (-2.34%)	0.87 (-3.13%)
	DeepFool	0.69	0.73 (-19.3%)	0.95 (-2.69%)	0.68 (-20.97%)	0.62 (-22.78%)	0.73 (-17.23%)	0.7 (-20.19%)

while tampering with just one signal field— with a minimum length of 9 bits— making the payload progressively easier to detect as the attack continues. Unfortunately, this behavior also causes the algorithms to strongly perturb the target signal, often resulting in an example that closely resembles a packet from the attack-free scenario, meaning that while the evasion is successful, the intended effect is lost. Notably, while DeepFool performs best in the fuzzy attack, in these cases it is the L2 BIM algorithm that consistently achieves a 40% to 50

The masquerade replay attack (see Table 7) justifiably has the lowest detection rates in the baselines, due to the attack signals being valid sequences in themselves, simply inserted at a different moment on the network by the attacker. Therefore, the system can only leverage the semantic discontinuity in the flow of messages for its classification. The FFNN algorithm is rendered completely useless, meaning no evasion is necessary or achieved. CANdito with the L2 BIM algorithm performs best in terms of detection loss, with a 23% reduction, but overall the evasion capabilities are relatively low, likely due to the already low detection performances of the detection systems.

Finally, in the injection replay scenario (see Table 8), the attacker interleaves additional replayed packets into the normal traffic so that the content is analogous to full replay sequences, but

Table 5. Results on the ReCAN dataset in the white-box, grey-box (first part of the table, the white diagonal line represent the white-box), and black-box scenarios for the continuous change attack.

White-box and Grey-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.02	0.2 (-23.64%)	0.78 (0.0%)	0.62 (-0.76%)	0.62 (-0.63%)	0.61 (-0.99%)	0.62 (-0.18%)
	L2 BIM	0.06	0.07 (-35.68%)	0.78 (0.0%)	0.6 (-3.41%)	0.58 (-4.33%)	0.59 (-2.86%)	0.6 (-2.37%)
	DeepFool	0.04	0.21 (-22.11%)	0.78 (0.0%)	0.62 (-0.86%)	0.61 (-1.68%)	0.62 (-0.1%)	0.62 (0.18%)
CANdito	Decay BIM	0.0	0.43 (0.0%)	0.78 (0.0%)	0.63 (0.0%)	0.63 (0.0%)	0.62 (0.0%)	0.62 (0.0%)
	L2 BIM	0.0	0.42 (-0.77%)	0.65 (-12.22%)	0.63 (-0.27%)	0.62 (-0.19%)	0.62 (-0.47%)	0.62 (-0.33%)
	DeepFool	0.02	0.4 (-3.54%)	0.68 (-9.74%)	0.62 (-0.8%)	0.61 (-1.98%)	0.61 (-0.84%)	0.61 (-0.8%)
Short LSTM	Decay BIM	0.02	0.39 (-4.2%)	0.74 (-3.57%)	0.45 (-18.54%)	0.55 (-7.9%)	0.54 (-8.34%)	0.53 (-9.27%)
	L2 BIM	0.13	0.26 (-17.17%)	0.65 (-12.76%)	0.17 (-46.27%)	0.32 (-30.91%)	0.38 (-23.96%)	0.35 (-27.31%)
	DeepFool	0.25	0.29 (-13.7%)	0.67 (-10.61%)	0.24 (-38.88%)	0.4 (-22.48%)	0.45 (-17.2%)	0.43 (-18.71%)
Long LSTM	Decay BIM	0.02	0.38 (-4.8%)	0.74 (-3.57%)	0.55 (-8.64%)	0.44 (-18.67%)	0.54 (-8.38%)	0.53 (-9.11%)
	L2 BIM	0.13	0.24 (-18.97%)	0.62 (-15.66%)	0.36 (-26.8%)	0.12 (-50.23%)	0.39 (-23.26%)	0.37 (-25.32%)
	DeepFool	0.22	0.3 (-13.52%)	0.73 (-4.81%)	0.43 (-19.93%)	0.23 (-39.94%)	0.49 (-13.69%)	0.46 (-15.47%)
Short GRU	Decay BIM	0.03	0.37 (-5.84%)	0.76 (-1.19%)	0.57 (-6.38%)	0.57 (-5.26%)	0.41 (-21.31%)	0.55 (-6.82%)
	L2 BIM	0.13	0.25 (-18.44%)	0.64 (-13.79%)	0.37 (-26.02%)	0.37 (-25.32%)	0.14 (-47.84%)	0.35 (-27.39%)
	DeepFool	0.25	0.31 (-12.29%)	0.67 (-10.27%)	0.42 (-20.76%)	0.41 (-21.23%)	0.23 (-39.7%)	0.45 (-17.23%)
Long GRU	Decay BIM	0.02	0.39 (-4.18%)	0.76 (-1.19%)	0.57 (-6.43%)	0.58 (-4.89%)	0.53 (-9.67%)	0.44 (-18.26%)
	L2 BIM	0.12	0.27 (-15.7%)	0.7 (-7.94%)	0.45 (-18.37%)	0.45 (-17.68%)	0.39 (-23.49%)	0.17 (-45.07%)
	DeepFool	0.26	0.31 (-11.86%)	0.68 (-9.51%)	0.43 (-20.02%)	0.43 (-19.79%)	0.43 (-19.44%)	0.22 (-39.44%)
Black-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.02	0.39 (-4.58%)	0.76 (-1.19%)	0.62 (-1.3%)	0.62 (-0.82%)	0.61 (-0.79%)	0.61 (-0.41%)
	L2 BIM	0.06	0.35 (-7.92%)	0.76 (-1.52%)	0.61 (-2.36%)	0.61 (-1.38%)	0.59 (-3.3%)	0.61 (-1.24%)
	DeepFool	0.04	0.4 (-2.83%)	0.77 (-0.76%)	0.62 (-1.17%)	0.62 (-0.83%)	0.61 (-1.64%)	0.61 (-0.66%)
CANdito	Decay BIM	0.0	0.43 (0.0%)	0.78 (0.0%)	0.63 (0.0%)	0.63 (0.0%)	0.62 (0.0%)	0.62 (0.0%)
	L2 BIM	0.0	0.43 (-0.24%)	0.76 (-1.76%)	0.63 (-0.06%)	0.63 (0.11%)	0.62 (-0.11%)	0.62 (0.06%)
	DeepFool	0.01	0.43 (-0.31%)	0.77 (-0.76%)	0.63 (-0.19%)	0.62 (-0.07%)	0.62 (-0.39%)	0.62 (-0.2%)
Short LSTM	Decay BIM	0.02	0.37 (-6.2%)	0.76 (-1.19%)	0.59 (-4.26%)	0.59 (-4.06%)	0.56 (-6.71%)	0.58 (-4.26%)
	L2 BIM	0.09	0.3 (-13.2%)	0.72 (-5.65%)	0.5 (-13.23%)	0.48 (-14.18%)	0.49 (-13.39%)	0.49 (-13.4%)
	DeepFool	0.25	0.31 (-12.42%)	0.71 (-7.07%)	0.47 (-16.58%)	0.46 (-16.88%)	0.48 (-14.17%)	0.49 (-13.19%)
Long LSTM	Decay BIM	0.02	0.37 (-6.58%)	0.76 (-1.19%)	0.59 (-4.08%)	0.59 (-3.62%)	0.56 (-5.74%)	0.58 (-3.99%)
	L2 BIM	0.09	0.27 (-16.09%)	0.71 (-6.7%)	0.48 (-14.89%)	0.48 (-14.52%)	0.49 (-13.32%)	0.48 (-13.57%)
	DeepFool	0.25	0.29 (-14.34%)	0.68 (-9.51%)	0.46 (-17.42%)	0.45 (-17.46%)	0.47 (-14.94%)	0.49 (-13.18%)
Short GRU	Decay BIM	0.03	0.36 (-6.98%)	0.73 (-4.09%)	0.64 (1.07%)	0.63 (0.4%)	0.63 (0.42%)	0.63 (1.2%)
	L2 BIM	0.09	0.32 (-11.33%)	0.74 (-3.88%)	0.57 (-6.44%)	0.56 (-6.73%)	0.53 (-9.28%)	0.56 (-6.14%)
	DeepFool	0.22	0.36 (-7.11%)	0.71 (-6.31%)	0.5 (-12.83%)	0.5 (-12.86%)	0.49 (-13.16%)	0.51 (-10.5%)
Long GRU	Decay BIM	0.02	0.36 (-6.91%)	0.76 (-1.19%)	0.59 (-3.81%)	0.6 (-2.72%)	0.57 (-5.74%)	0.59 (-3.28%)
	L2 BIM	0.09	0.31 (-12.34%)	0.74 (-3.27%)	0.54 (-8.9%)	0.54 (-8.09%)	0.5 (-12.06%)	0.52 (-9.81%)
	DeepFool	0.24	0.3 (-13.1%)	0.69 (-8.56%)	0.47 (-16.04%)	0.47 (-15.86%)	0.48 (-14.02%)	0.48 (-13.64%)

discontinuities are present at each injection. This is the only attack against which the predictors outperform CANdito in the baseline test. Similarly, the drop in detection rate is more pronounced in the predictor models than in CANdito. As in the previous case, since it does not consider temporal windows for detection, the FFNN is rendered completely useless.

CarHacking. Interestingly, the results on the dataset are heavily dependent on the evasion algorithm used. In fact, the DeepFool algorithm is the only one to achieve any kind of evasion in both the gear and RPM spoofing attack scenarios (see Tables 10 and 9). In these attacks, however, the drop in detection performance is up to 85% for the FFNN and remains consistently high for all but CANdito, with the only exception being the Long GRU model in the RPM spoofing attack.

7.4.3 Grey-Box scenario. In this experiment, we compare the performances of all six IDSs over all the available attacks in the grey-box scenario. Tables, 4, 5, 6, 7, and 8 show the aggregated results for the adversarial attacks on the ReCAN dataset, while Tables 9 and 10 on the CarHacking dataset. We remember the reader that since we test the evasive packets generated by each oracle model against all the other available architectures on the same dataset, the diagonal in these tables represents the results of the white-box scenario, while the remaining part the results of the grey-box one.

Table 6. Results on the ReCAN dataset in the white-box, grey-box (first part of the table, the white diagonal line represent the white-box), and black-box scenarios for the change to minimum attack.

White-box and Grey-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.02	0.31 (-20.7%)	0.85 (6.49%)	0.65 (12.43%)	0.68 (12.09%)	0.68 (15.3%)	0.67 (16.03%)
	L2 BIM	0.09	0.08 (-43.68%)	0.81 (2.88%)	0.61 (8.13%)	0.61 (5.56%)	0.61 (9.24%)	0.63 (11.4%)
	DeepFool	0.06	0.27 (-24.89%)	0.85 (6.8%)	0.63 (10.27%)	0.64 (8.68%)	0.64 (12.26%)	0.64 (12.87%)
CANdito	Decay BIM	0.0	0.56 (4.71%)	0.78 (0.0%)	0.67 (13.87%)	0.68 (12.26%)	0.68 (16.18%)	0.68 (16.82%)
	L2 BIM	0.0	0.53 (1.22%)	0.72 (-6.89%)	0.66 (12.87%)	0.67 (11.87%)	0.68 (15.49%)	0.67 (16.32%)
	DeepFool	0.01	0.52 (0.91%)	0.76 (-2.26%)	0.65 (12.22%)	0.66 (10.21%)	0.67 (14.53%)	0.66 (15.17%)
Short LSTM	Decay BIM	0.03	0.5 (-1.09%)	0.86 (7.95%)	0.46 (-6.9%)	0.6 (4.99%)	0.6 (7.65%)	0.6 (8.42%)
	L2 BIM	0.15	0.34 (-17.89%)	0.69 (-9.82%)	0.18 (-34.66%)	0.38 (-17.26%)	0.4 (-11.91%)	0.39 (-12.6%)
	DeepFool	0.26	0.35 (-16.6%)	0.74 (-4.48%)	0.21 (-32.34%)	0.43 (-12.26%)	0.47 (-4.85%)	0.44 (-6.68%)
Long LSTM	Decay BIM	0.03	0.5 (-1.59%)	0.88 (9.46%)	0.58 (5.32%)	0.47 (-8.02%)	0.6 (7.39%)	0.59 (8.14%)
	L2 BIM	0.15	0.32 (-20.07%)	0.71 (-7.87%)	0.33 (-19.96%)	0.15 (-40.26%)	0.38 (-14.19%)	0.35 (-15.91%)
	DeepFool	0.19	0.38 (-13.35%)	0.85 (6.05%)	0.47 (-6.19%)	0.27 (-28.36%)	0.54 (1.84%)	0.52 (0.47%)
Short GRU	Decay BIM	0.03	0.5 (-1.52%)	0.89 (10.51%)	0.6 (7.06%)	0.64 (8.26%)	0.48 (-4.5%)	0.61 (10.03%)
	L2 BIM	0.14	0.35 (-16.31%)	0.75 (-3.36%)	0.42 (-10.93%)	0.45 (-10.92%)	0.19 (-33.1%)	0.41 (-10.13%)
	DeepFool	0.25	0.37 (-14.24%)	0.75 (-2.95%)	0.43 (-9.82%)	0.45 (-10.65%)	0.25 (-26.92%)	0.47 (-4.37%)
Long GRU	Decay BIM	0.02	0.49 (-2.24%)	0.88 (9.46%)	0.6 (6.51%)	0.62 (7.0%)	0.59 (6.52%)	0.51 (0.0%)
	L2 BIM	0.15	0.35 (-16.9%)	0.74 (-4.75%)	0.42 (-10.78%)	0.46 (-9.48%)	0.4 (-12.37%)	0.19 (-32.3%)
	DeepFool	0.26	0.35 (-16.3%)	0.72 (-6.37%)	0.45 (-8.07%)	0.47 (-8.6%)	0.45 (-6.87%)	0.24 (-27.08%)
Black-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.02	0.28 (-23.68%)	0.75 (-2.99%)	0.55 (2.24%)	0.58 (2.65%)	0.54 (1.94%)	0.53 (2.13%)
	L2 BIM	0.08	0.08 (-43.12%)	0.57 (-21.14%)	0.53 (-0.09%)	0.56 (0.74%)	0.5 (-2.22%)	0.52 (0.43%)
	DeepFool	0.04	0.27 (-24.39%)	0.72 (-6.26%)	0.55 (1.74%)	0.58 (2.3%)	0.53 (1.15%)	0.53 (2.04%)
CANdito	Decay BIM	0.01	0.51 (-0.44%)	0.68 (-10.22%)	0.56 (3.24%)	0.58 (2.92%)	0.55 (3.2%)	0.55 (3.9%)
	L2 BIM	0.0	0.5 (-1.39%)	0.58 (-20.34%)	0.56 (2.91%)	0.58 (2.74%)	0.55 (2.86%)	0.55 (3.49%)
	DeepFool	0.0	0.51 (-0.86%)	0.64 (-14.36%)	0.56 (2.71%)	0.58 (2.86%)	0.55 (2.8%)	0.54 (3.17%)
Short LSTM	Decay BIM	0.02	0.44 (-7.97%)	0.74 (-4.03%)	0.4 (-12.97%)	0.52 (-3.55%)	0.46 (-5.86%)	0.45 (-5.83%)
	L2 BIM	0.1	0.32 (-19.3%)	0.66 (-12.39%)	0.18 (-35.13%)	0.37 (-18.43%)	0.33 (-19.06%)	0.31 (-20.35%)
	DeepFool	0.23	0.33 (-18.65%)	0.65 (-13.7%)	0.16 (-37.07%)	0.33 (-22.13%)	0.33 (-18.96%)	0.31 (-19.71%)
Long LSTM	Decay BIM	0.02	0.45 (-6.84%)	0.73 (-5.22%)	0.49 (-4.16%)	0.41 (-14.79%)	0.47 (-4.91%)	0.47 (-3.89%)
	L2 BIM	0.11	0.3 (-21.9%)	0.63 (-15.52%)	0.32 (-21.03%)	0.18 (-37.8%)	0.32 (-20.17%)	0.31 (-20.15%)
	DeepFool	0.25	0.31 (-20.41%)	0.58 (-20.88%)	0.35 (-17.93%)	0.13 (-42.05%)	0.31 (-21.44%)	0.31 (-20.36%)
Short GRU	Decay BIM	0.02	0.46 (-5.41%)	0.75 (-2.99%)	0.52 (-0.83%)	0.55 (-0.55%)	0.37 (-15.11%)	0.48 (-3.25%)
	L2 BIM	0.11	0.35 (-16.29%)	0.67 (-11.5%)	0.34 (-19.12%)	0.39 (-16.08%)	0.16 (-36.69%)	0.31 (-20.25%)
	DeepFool	0.22	0.39 (-12.62%)	0.64 (-14.67%)	0.37 (-16.02%)	0.39 (-16.49%)	0.18 (-34.47%)	0.36 (-15.66%)
Long GRU	Decay BIM	0.02	0.44 (-8.05%)	0.75 (-2.99%)	0.52 (-1.11%)	0.54 (-1.38%)	0.47 (-5.57%)	0.39 (-12.65%)
	L2 BIM	0.11	0.33 (-18.51%)	0.7 (-8.08%)	0.34 (-19.47%)	0.36 (-19.0%)	0.31 (-21.14%)	0.15 (-36.41%)
	DeepFool	0.26	0.32 (-19.64%)	0.6 (-17.96%)	0.36 (-17.17%)	0.35 (-20.43%)	0.33 (-19.62%)	0.14 (-37.25%)

ReCAN. We refrain from evaluating each attack scenario again due to repetitiveness and space constraints and instead highlight the most notable characteristics of the grey-box performances. Overall, the results are, as expected, worse than in the white-box scenario and better than in the black-box one. In some attack scenarios where the white-box attacks are effective, such as the fuzzy and change-to-minimum attacks, the performance of using oracles different from the detection model, although trained on the same data, drops significantly. The delta in performance loss is less prominent in the already least-performing attacks. Probably the most notable characteristic of the grey-box scenario is that the autoencoder models, namely CANdito and the FFNN, when used to attack the predictive models, perform poorly enough to even increase the detection performance, as is the case with the change-to-minimum attack. On the other hand, the FFNN used as a defense system seems to increase its detection rate in grey-box attacks in the masquerade and injection replay scenarios, mostly because these perturbations sometimes alter the shape of the packet from one identical to previous traffic to one different enough to be detected. Nevertheless, the overall detection performances of the algorithm in such scenarios still render it ineffective.

Table 7. Results on the ReCAN dataset in the white-box, grey-box (first part of the table, the white diagonal line represent the white-box), and black-box scenarios for the masquerade replay attack.

White-box and Grey-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.0	0.0 (-0.06%)	0.36 (0.0%)	0.27 (0.0%)	0.26 (0.0%)	0.21 (0.0%)	0.22 (0.0%)
	L2 BIM	0.0	0.0 (-0.06%)	0.36 (0.0%)	0.27 (0.0%)	0.26 (0.0%)	0.21 (0.0%)	0.22 (0.0%)
	DeepFool	0.0	0.0 (-0.06%)	0.36 (0.0%)	0.27 (0.0%)	0.26 (0.0%)	0.21 (0.0%)	0.22 (0.0%)
CANdito	Decay BIM	0.01	0.01 (0.56%)	0.24 (-12.41%)	0.28 (0.96%)	0.27 (1.09%)	0.22 (1.21%)	0.23 (0.53%)
	L2 BIM	0.01	0.0 (0.19%)	0.12 (-23.7%)	0.29 (1.2%)	0.26 (0.87%)	0.21 (0.09%)	0.23 (0.49%)
	DeepFool	0.0	0.0 (0.06%)	0.25 (-10.65%)	0.29 (1.22%)	0.27 (1.06%)	0.21 (0.39%)	0.23 (0.57%)
Short LSTM	Decay BIM	0.02	0.01 (1.18%)	0.3 (-5.93%)	0.14 (-13.17%)	0.23 (-2.89%)	0.19 (-1.46%)	0.22 (-0.16%)
	L2 BIM	0.06	0.02 (2.33%)	0.27 (-8.54%)	0.09 (-18.62%)	0.13 (-12.41%)	0.14 (-6.61%)	0.15 (-7.57%)
	DeepFool	0.16	0.01 (0.97%)	0.29 (-6.6%)	0.09 (-18.8%)	0.15 (-11.06%)	0.13 (-7.4%)	0.13 (-9.63%)
Long LSTM	Decay BIM	0.02	0.01 (0.99%)	0.31 (-4.88%)	0.21 (-5.96%)	0.14 (-11.64%)	0.18 (-2.67%)	0.2 (-2.41%)
	L2 BIM	0.06	0.02 (1.53%)	0.3 (-5.83%)	0.14 (-13.89%)	0.09 (-17.03%)	0.14 (-6.91%)	0.15 (-7.41%)
	DeepFool	0.15	0.01 (0.47%)	0.34 (-1.97%)	0.13 (-14.86%)	0.07 (-18.91%)	0.12 (-8.87%)	0.13 (-9.37%)
Short GRU	Decay BIM	0.02	0.01 (0.96%)	0.31 (-4.88%)	0.22 (-5.1%)	0.22 (-3.51%)	0.12 (-8.66%)	0.2 (-2.44%)
	L2 BIM	0.05	0.02 (2.04%)	0.27 (-9.35%)	0.19 (-8.38%)	0.2 (-5.89%)	0.08 (-12.46%)	0.15 (-7.4%)
	DeepFool	0.15	0.0 (0.36%)	0.31 (-4.63%)	0.16 (-11.21%)	0.15 (-10.79%)	0.08 (-12.33%)	0.12 (-9.79%)
Long GRU	Decay BIM	0.02	0.02 (1.86%)	0.32 (-3.84%)	0.23 (-4.64%)	0.22 (-3.63%)	0.19 (-1.84%)	0.12 (-10.22%)
	L2 BIM	0.05	0.03 (2.84%)	0.3 (-6.46%)	0.2 (-7.62%)	0.21 (-4.78%)	0.15 (-6.17%)	0.08 (-14.0%)
	DeepFool	0.15	0.0 (0.12%)	0.32 (-4.51%)	0.16 (-11.36%)	0.14 (-11.18%)	0.12 (-8.83%)	0.08 (-14.39%)
Black-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.0	0.0 (0.0%)	0.36 (0.0%)	0.27 (0.0%)	0.26 (0.0%)	0.21 (0.0%)	0.22 (0.0%)
	L2 BIM	0.0	0.0 (0.0%)	0.36 (0.0%)	0.27 (-0.0%)	0.26 (0.0%)	0.21 (0.0%)	0.22 (-0.0%)
	DeepFool	0.0	0.0 (0.0%)	0.36 (0.0%)	0.27 (0.0%)	0.26 (0.0%)	0.21 (0.0%)	0.22 (0.0%)
CANdito	Decay BIM	0.0	0.01 (0.47%)	0.36 (0.0%)	0.29 (1.3%)	0.27 (1.7%)	0.22 (1.3%)	0.24 (1.77%)
	L2 BIM	0.0	0.0 (0.07%)	0.33 (-2.84%)	0.28 (0.59%)	0.26 (0.08%)	0.21 (-0.09%)	0.22 (0.1%)
	DeepFool	0.0	0.0 (0.07%)	0.34 (-2.08%)	0.28 (0.76%)	0.26 (0.47%)	0.21 (0.02%)	0.23 (0.39%)
Short LSTM	Decay BIM	0.01	0.01 (0.74%)	0.31 (-5.09%)	0.24 (-3.63%)	0.22 (-3.98%)	0.19 (-1.39%)	0.2 (-1.82%)
	L2 BIM	0.04	0.0 (0.28%)	0.29 (-7.18%)	0.22 (-5.82%)	0.2 (-5.91%)	0.18 (-3.11%)	0.18 (-4.12%)
	DeepFool	0.14	0.01 (0.63%)	0.32 (-3.7%)	0.17 (-10.38%)	0.16 (-10.06%)	0.14 (-7.19%)	0.15 (-7.48%)
Long LSTM	Decay BIM	0.01	0.02 (1.46%)	0.34 (-2.29%)	0.24 (-2.96%)	0.22 (-3.63%)	0.21 (-0.12%)	0.21 (-0.74%)
	L2 BIM	0.04	0.0 (0.08%)	0.29 (-7.38%)	0.2 (-7.33%)	0.17 (-8.1%)	0.17 (-3.36%)	0.18 (-4.44%)
	DeepFool	0.14	0.0 (0.34%)	0.31 (-4.63%)	0.17 (-10.92%)	0.14 (-11.09%)	0.13 (-7.56%)	0.13 (-9.06%)
Short GRU	Decay BIM	0.02	0.02 (2.31%)	0.3 (-6.15%)	0.25 (-2.43%)	0.23 (-2.6%)	0.2 (-1.13%)	0.21 (-0.7%)
	L2 BIM	0.04	0.04 (4.24%)	0.33 (-3.33%)	0.25 (-2.07%)	0.23 (-2.2%)	0.21 (0.1%)	0.23 (0.6%)
	DeepFool	0.15	0.01 (0.73%)	0.3 (-5.56%)	0.17 (-10.06%)	0.15 (-10.32%)	0.14 (-6.94%)	0.14 (-8.01%)
Long GRU	Decay BIM	0.02	0.01 (1.13%)	0.31 (-5.3%)	0.25 (-2.79%)	0.23 (-2.92%)	0.2 (-0.63%)	0.21 (-0.79%)
	L2 BIM	0.04	0.02 (2.07%)	0.32 (-4.38%)	0.22 (-5.78%)	0.2 (-5.72%)	0.17 (-3.24%)	0.19 (-3.31%)
	DeepFool	0.15	0.01 (0.52%)	0.31 (-4.63%)	0.17 (-10.03%)	0.15 (-10.32%)	0.14 (-6.88%)	0.14 (-8.18%)

CarHacking. Aside from the lack of effectiveness of the BIM algorithms, the grey-box scenarios for the CarHacking dataset mostly confirm what was already observed in the white-box scenario. With DeepFool, the FFNN is extremely effective against most models, while the other oracles lose some effectiveness but still maintain evasion capabilities. Interestingly, the FFNN also proves to be effective as an intrusion detection system, apparently contradicting the results from the ReCAN dataset. The hypothesis for why this occurs is that the ease of detection for the FFNN (and partially for all detection systems) stems from the simplicity of the signal characteristics in the CarHacking dataset, which are almost always constant. Notably, CANdito is the only IDS that is not evaded by any oracle, but it is also the only oracle that does not find any suitable evasive point.

7.4.4 Black-Box scenario. Finally, we compare the performances of all six IDSs over all the available attacks in the black-box scenario. Tables, 4, 5, 6, 7, and 8 show the aggregated results for the adversarial attacks on the ReCAN dataset, while Tables 9 and 10 on the CarHacking dataset. The second part of the table represents the black-box scenario. It is interesting to highlight that the diagonal line of the black-box tables represents the case where the oracle and IDS share the

Table 8. Results on the ReCAN dataset in the white-box, grey-box (first part of the table, the white diagonal line represent the white-box), and black-box scenarios for the injection replay attack.

White-box and Grey-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.0	0.0 (0.0%)	0.42 (0.0%)	0.66 (0.0%)	0.54 (0.0%)	0.69 (0.0%)	0.69 (0.0%)
	L2 BIM	0.0	0.0 (0.0%)	0.42 (0.0%)	0.66 (0.0%)	0.54 (0.0%)	0.69 (0.0%)	0.69 (0.0%)
	DeepFool	0.0	0.0 (0.0%)	0.42 (0.0%)	0.66 (0.0%)	0.54 (0.0%)	0.69 (0.0%)	0.69 (0.0%)
CANdito	Decay BIM	0.0	0.0 (0.0%)	0.41 (-1.01%)	0.66 (0.0%)	0.54 (0.0%)	0.69 (0.0%)	0.69 (0.0%)
	L2 BIM	0.01	0.0 (0.07%)	0.29 (-13.5%)	0.66 (-0.6%)	0.54 (-0.73%)	0.69 (-0.2%)	0.68 (-0.8%)
	DeepFool	0.0	0.0 (0.0%)	0.39 (-3.18%)	0.66 (0.0%)	0.54 (0.0%)	0.69 (0.0%)	0.69 (0.0%)
Short LSTM	Decay BIM	0.03	0.01 (0.87%)	0.34 (-8.64%)	0.36 (-30.47%)	0.38 (-16.2%)	0.53 (-16.53%)	0.49 (-19.6%)
	L2 BIM	0.08	0.03 (3.2%)	0.21 (-20.99%)	0.21 (-45.13%)	0.24 (-30.27%)	0.38 (-31.73%)	0.35 (-34.0%)
	DeepFool	0.19	0.03 (2.6%)	0.46 (4.3%)	0.24 (-42.07%)	0.3 (-24.6%)	0.52 (-17.6%)	0.53 (-16.2%)
Long LSTM	Decay BIM	0.03	0.03 (3.07%)	0.35 (-6.81%)	0.51 (-15.33%)	0.32 (-22.93%)	0.54 (-15.67%)	0.53 (-15.6%)
	L2 BIM	0.08	0.03 (3.27%)	0.24 (-18.6%)	0.41 (-25.13%)	0.16 (-38.33%)	0.44 (-24.93%)	0.44 (-24.87%)
	DeepFool	0.19	0.04 (4.47%)	0.45 (3.02%)	0.48 (-18.73%)	0.16 (-38.13%)	0.52 (-17.07%)	0.51 (-17.6%)
Short GRU	Decay BIM	0.03	0.02 (1.53%)	0.34 (-8.12%)	0.43 (-23.2%)	0.4 (-14.47%)	0.38 (-31.07%)	0.43 (-25.53%)
	L2 BIM	0.08	0.03 (3.33%)	0.21 (-20.99%)	0.33 (-33.0%)	0.29 (-25.8%)	0.22 (-46.8%)	0.3 (-39.0%)
	DeepFool	0.19	0.04 (4.07%)	0.43 (0.83%)	0.46 (-20.33%)	0.37 (-17.47%)	0.28 (-41.13%)	0.47 (-22.33%)
Long GRU	Decay BIM	0.03	0.01 (1.4%)	0.34 (-8.64%)	0.42 (-24.87%)	0.41 (-13.73%)	0.5 (-19.73%)	0.38 (-31.4%)
	L2 BIM	0.08	0.04 (3.6%)	0.21 (-20.99%)	0.31 (-35.6%)	0.27 (-27.13%)	0.31 (-38.07%)	0.22 (-46.67%)
	DeepFool	0.19	0.04 (3.67%)	0.39 (-2.77%)	0.42 (-24.67%)	0.34 (-20.33%)	0.5 (-19.6%)	0.26 (-43.0%)
Black-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.0	0.0 (0.0%)	0.42 (0.0%)	0.66 (0.0%)	0.54 (0.0%)	0.69 (0.0%)	0.69 (0.0%)
	L2 BIM	0.0	0.0 (0.0%)	0.42 (0.0%)	0.66 (0.0%)	0.54 (0.0%)	0.69 (0.0%)	0.69 (0.0%)
	DeepFool	0.0	0.0 (0.0%)	0.42 (0.0%)	0.66 (0.0%)	0.54 (0.0%)	0.69 (0.0%)	0.69 (0.0%)
CANdito	Decay BIM	0.0	0.01 (0.6%)	0.42 (-0.52%)	0.66 (-0.07%)	0.54 (-0.07%)	0.69 (-0.13%)	0.69 (0.0%)
	L2 BIM	0.0	0.0 (0.07%)	0.38 (-4.6%)	0.65 (-1.13%)	0.53 (-1.53%)	0.68 (-0.8%)	0.67 (-1.6%)
	DeepFool	0.0	0.0 (0.0%)	0.42 (-0.52%)	0.66 (0.0%)	0.54 (0.0%)	0.69 (0.0%)	0.69 (0.0%)
Short LSTM	Decay BIM	0.03	0.01 (0.67%)	0.33 (-9.62%)	0.5 (-16.33%)	0.47 (-7.67%)	0.61 (-8.6%)	0.54 (-14.87%)
	L2 BIM	0.05	0.02 (2.2%)	0.27 (-14.88%)	0.47 (-19.8%)	0.42 (-12.0%)	0.52 (-16.8%)	0.49 (-20.33%)
	DeepFool	0.17	0.02 (2.2%)	0.45 (2.57%)	0.48 (-18.33%)	0.38 (-16.27%)	0.57 (-12.07%)	0.54 (-15.33%)
Long LSTM	Decay BIM	0.03	0.02 (1.93%)	0.33 (-9.62%)	0.53 (-12.93%)	0.46 (-8.87%)	0.61 (-7.8%)	0.58 (-10.93%)
	L2 BIM	0.05	0.03 (3.27%)	0.27 (-14.88%)	0.48 (-18.0%)	0.41 (-13.13%)	0.56 (-13.6%)	0.54 (-15.33%)
	DeepFool	0.18	0.03 (3.47%)	0.44 (1.67%)	0.51 (-15.6%)	0.39 (-15.87%)	0.55 (-14.2%)	0.56 (-12.73%)
Short GRU	Decay BIM	0.03	0.01 (1.1%)	0.32 (-10.29%)	0.5 (-16.23%)	0.47 (-7.72%)	0.59 (-10.71%)	0.55 (-13.73%)
	L2 BIM	0.06	0.05 (4.8%)	0.29 (-12.66%)	0.46 (-20.6%)	0.43 (-11.73%)	0.49 (-20.73%)	0.5 (-19.27%)
	DeepFool	0.2	0.07 (6.87%)	0.46 (3.46%)	0.48 (-18.6%)	0.39 (-15.47%)	0.52 (-16.8%)	0.54 (-15.0%)
Long GRU	Decay BIM	0.03	0.02 (2.33%)	0.34 (-8.51%)	0.5 (-16.13%)	0.46 (-8.53%)	0.58 (-10.8%)	0.56 (-12.87%)
	L2 BIM	0.05	0.05 (5.0%)	0.29 (-13.22%)	0.47 (-19.73%)	0.42 (-12.27%)	0.52 (-17.2%)	0.5 (-19.2%)
	DeepFool	0.19	0.04 (4.47%)	0.44 (1.66%)	0.5 (-16.4%)	0.39 (-15.73%)	0.54 (-15.53%)	0.53 (-15.8%)

same architecture, but the models have been trained on different datasets. However, the identical architecture does not appear to provide a noticeable increase in evasion capabilities as long as the training data is different.

ReCAN. As expected, the performances of the oracles in the black-box scenario are significantly lower than in the white-box scenario. However, the DeepFool algorithm maintains a consistent 20% drop in detection rate for the predictive models in all attacks except the masquerade injection case. In the black-box scenario, the predictive models remain the overall better oracles, consistently achieving better evasion performances than the autoencoders, with both the FFNN and CANdito consistently showing less than a 10% loss in detection rate. Nonetheless, in terms of detection system performance, CANdito remains the hardest to evade, often experiencing a 5-20% smaller drop in detection rate than the other detectors.

CarHacking. The results in the black-box scenario for the CarHacking dataset appear similar to those in the white- and grey-box scenarios, with the predictive oracles being able to find evasive points only against predictive IDSs, and CANdito achieving a perfect TPR against all oracles.

Table 9. Results on the CarHacking dataset in the white-box, grey-box (first part of the table, the white diagonal line represent the white-box), and black-box scenarios for the RPM spoofing attack.

White-box and Grey-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.68	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	0.15 (-85.02%)	1.0 (0.0%)	0.15 (-84.51%)	0.15 (-84.51%)	0.15 (-84.51%)	0.15 (-84.51%)
CANdito	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
Short LSTM	Decay BIM	0.03	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.01	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	0.68 (-32.13%)	1.0 (0.0%)	0.97 (-2.99%)	0.99 (-1.14%)
Long LSTM	Decay BIM	0.02	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.02	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.01	1.0 (0.0%)	1.0 (0.0%)	0.88 (-11.53%)	0.75 (-24.91%)	0.85 (-14.7%)	0.7 (-29.84%)
Short GRU	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	0.85 (-15.49%)	1.0 (0.0%)
Long GRU	Decay BIM	0.02	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.02	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.01	1.0 (0.0%)	1.0 (0.0%)	0.99 (-0.88%)	0.99 (-0.88%)	0.99 (-1.06%)	0.99 (-1.5%)
Black-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	0.15 (-85.02%)	1.0 (0.0%)	0.15 (-84.51%)	0.15 (-84.51%)	0.15 (-84.51%)	0.15 (-84.51%)
CANdito	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
Short LSTM	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	0.56 (-44.1%)	0.56 (-44.1%)	0.56 (-44.01%)	0.56 (-44.1%)
Long LSTM	Decay BIM	0.11	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.07	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	0.54 (-46.48%)	0.54 (-46.48%)	0.55 (-45.16%)	0.54 (-45.86%)
Short GRU	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	0.7 (-30.37%)	0.7 (-30.37%)	0.7 (-29.84%)	0.7 (-30.37%)
Long GRU	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	0.98 (-1.67%)	0.98 (-1.67%)	0.98 (-1.67%)	0.98 (-1.67%)

However, the FFNN performs exceptionally well as an oracle against all but CANdito in the gear spoofing attack, while it fails completely in the RPM spoofing attack.

7.4.5 Discussion on results. Tables 11 and 12 provide aggregated results of the performances of the various evasive algorithms, oracles, and IDSs in all scenarios and over all the tested attacks.

It is evident from Table 11 that the difference in performance between the white-box and the grey-/black-box scenarios suggests that knowing the exact model is significantly more effective than having access only to the training dataset, which appears to be closer to not having any information at all. The table also shows that DeepFool is capable of achieving much higher perturbations without being detected, making it the most effective evasion algorithm (note that while the lack of evasion capabilities of the BIM algorithms on the CarHacking dataset skews the results toward DeepFool, similar although attenuated results would be observed if only the ReCAN dataset were considered).

Table 12 combines the results across IDSs and oracles, highlighting that while CANdito, as expected, remains the most effective intrusion detection algorithm across all scenarios, it is also the worst-performing oracle, with even a slight average increase in detection in the grey-box scenario.

Table 10. Results on the CarHacking dataset in the white-box, grey-box (first part of the table, the white diagonal line represent the white-box), and black-box scenarios for the gear spoofing attack.

White-box and Grey-box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	0.17 (-83.46%)	0.14 (-86.0%)	0.17 (-82.9%)	0.17 (-82.9%)	0.17 (-82.9%)	0.17 (-82.9%)
CANdito	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
Short LSTM	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.2	1.0 (0.0%)	1.0 (0.0%)	0.6 (-40.07%)	0.85 (-15.2%)	0.56 (-44.13%)	0.75 (-25.04%)
Long LSTM	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.05	1.0 (-0.25%)	1.0 (0.0%)	0.49 (-51.04%)	0.46 (-54.4%)	0.61 (-38.77%)	0.96 (-3.8%)
Short GRU	Decay BIM	0.04	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.04	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.01	0.73 (-26.73%)	1.0 (0.0%)	0.82 (-17.79%)	0.95 (-4.92%)	0.71 (-28.84%)	0.95 (-5.27%)
Long GRU	Decay BIM	0.1	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.12	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.02	1.0 (0.0%)	1.0 (0.0%)	1.0 (-0.09%)	1.0 (-0.09%)	1.0 (0.0%)	0.2 (-80.14%)
Black Box			TPR (Δ)					
Oracle	**Algorithm**	**AP**	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
FFNN	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
CANdito	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
Short LSTM	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (-0.17%)	1.0 (0.0%)	1.0 (-0.09%)	1.0 (0.0%)
Long LSTM	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (-0.09%)	1.0 (0.0%)	0.99 (-1.21%)	1.0 (0.0%)
Short GRU	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	0.86 (-13.53%)	1.0 (0.0%)	0.98 (-1.64%)	0.94 (-6.3%)	0.99 (-0.69%)	0.96 (-3.89%)
Long GRU	Decay BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	L2 BIM	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)
	DeepFool	0.0	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)	1.0 (0.0%)

Table 11. Average perturbation and delta TPR of the different evasive algorithms depending on the scenario.

	White Box			**Grey Box**			**Black Box**		
	Decay BIM	L2 BIM	DeepFool	Decay BIM	L2 BIM	DeepFool	Decay BIM	L2 BIM	DeepFool
Avg. Perturbation	0.02	0.08	0.17	0.02	0.08	0.17	0.01	0.05	0.15
Avg. Delta Recall	-7.76%	-20.13%	-30.88%	-0.82%	-5.71%	-10.44%	-2.02%	-5.21%	-10.41%

Overall, the predictor architectures are significantly more effective at achieving evasive samples, with the Short LSTM model performing best in all but the black-box scenario.

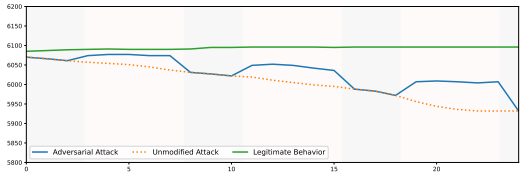
7.5 Experiment 2: Attack Perturbation

Figure 2 shows some exemplary adversarial behaviors to provide a qualitative assessment of the effectivness of the adversarially perturbed attacks on the vehicle by comparing the shape of the original with the adversarial attacks signals In the plot. In the figures, the dotted lines represent the intended content of an injected sequence while the blue lines are the results of the adversarial perturbation and the green lines provide a baseline reference depicting the normal signal in the attack-free state; a red background highlights packets that have successfully evaded the IDS and a grey background indicates the packets that were already undetected. We present

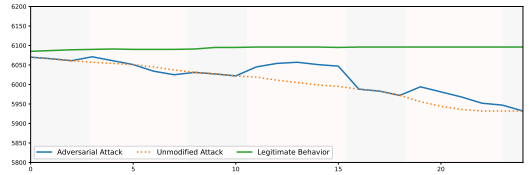
Table 12. Combined Results of IDSs and Oracles for Whitebox, Greybox, and Blackbox Scenarios.

Average IDS performances over all attacks						
Scenario	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
White Box	0.35	0.71	0.47	0.45	0.49	0.48
Grey Box	0.52	0.72	0.64	0.62	0.64	0.64
Black Box	0.51	0.73	0.64	0.62	0.64	0.64

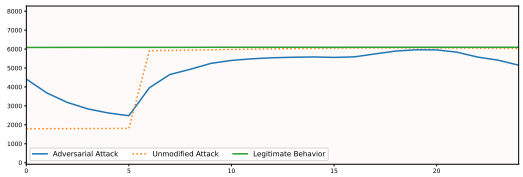
Average oracle TPR loss over all attacks						
Scenario	**FFNN**	**CANdito**	**Short LSTM**	**Long LSTM**	**Short GRU**	**Long GRU**
White Box	-20.63%	-4.61%	-23.97%	-23.86%	-21.31%	-23.15%
Grey Box	-6.02%	1.63%	-7.99%	-7.70%	-7.15%	-6.72%
Black Box	-4.64%	-0.19%	-8.31%	-8.63%	-6.94%	-6.58%



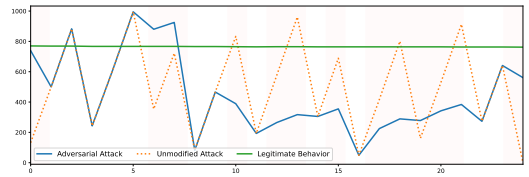
(a) White-box, Short LSTM oracle, Masquerade replay attack, DeepFool evasion algorithm, ID 0DE.



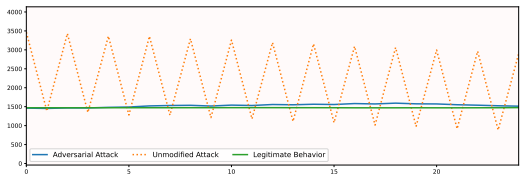
(b) Black-box, Short LSTM oracle, Masquerade replay attack, DeepFool evasion algorithm, ID 0DE.



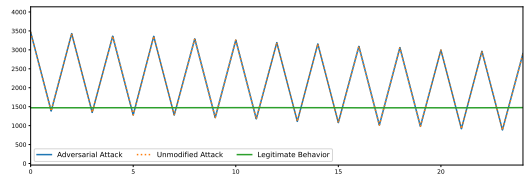
(c) White-box, Short GRU oracle, Continuous change attack, DeepFool evasion algorithm, ID 0DE.



(d) White-box, Short GRU oracle, Continuous change attack, DeepFool evasion algorithm, ID 0FF.



(e) Black-box, Short GRU oracle, Continuous change attack, L2 BIM evasion algorithm, ID 11C.



(f) Black-box, Short GRU oracle, Continuous change attack, DeepFool evasion algorithm, ID 11C.

Fig. 2. Example plots of attack events in various scenarios. In each plot the green line shows the untampered legitimate behavior of the signal, the dotted yellow line shows the basic version of the attack, and the blue line shows the evasive version of the attack.

three recurring scenarios worth discussing, cases in which the perturbation is effective, cases in which the perturbation nullifies the attacker goals, and cases in which the attack fails.

Plots 2a, 2b, and 2c. illustrate attack sequences that successfully achieve the initial attack goal while fully evading detection. It's worth noting, though unsurprising, that effective evasion strategies tend to involve staying close to the baseline reference (Plots 2a and 2b) and avoiding overly aggressive value changes (Plot 2c). Plots 2a and 2b depict nearly identical setups, with the key difference being the white-box vs. black-box scenario. Interestingly, the behavior of effective evasive sequences in both scenarios is often quite similar, as seen in this case. The main difference in evasion rate and perturbation, as previously shown in Table 11, is typically due to the algorithm's failure to find an

evasive perturbation, rather than any notable variation in the behavior of the generated evasive sequences themselves.

Plot 2e. captures an output sequence of the continuous change attack against the short GRU model for CAN ID "11C". There are multiple instances, throughout the experiments, of attacks that evade in a similar pattern, where the evasive points are very close to the reference normal traffic that the ECU would have transmitted if it was not silenced. This is particularly true for the two continuous experiments where the attack heavily manipulates only one specific signal and replays the others: the adversarial gradient ascent correctly captures the intra-packet dependency and pushes the rogue signal to values that are consistent with the context. This behavior is of course undesirable for a malicious actor since, despite the success of the evasion attempt, the meaning of the target payload is completely lost and the result is almost identical to not carrying out any attack. Note that not all evasive algorithms act similarly when attempting to generate an evasive attack, as visible in Plots 2e and 2f. Although both plots represent the same attack instance, scenario, and oracle, while the L2 BIM algorithm finds an evasive (but ineffective) perturbation, DeepFool fails.

Finally, plots 2d and 2f. depict perturbation scenarios that can be considered failures. In Plot 2d, the attack signal's unpredictable behavior prevents the algorithm from identifying a suitable perturbation for several packets in the sequence. In most of the evaded instances, the perturbed signal does not reach values similar to the original attack, casting doubt on the attack's overall effectiveness. If we focus on the correctly flagged packets that the algorithm failed to modify (shown with a white background), it becomes evident that it is easier to perturb the signal toward values resembling the last received packet (note that this behavior was not seen in CANdito, as the initial fully connected layer and target sequence reversal prevent the model from overly weighting the most recent packets during reconstruction, while also using a non-overlapping sliding window input). In Plot 2f, the algorithm completely fails to find a single evasive point throughout the entire sequence. It is worth noting that the algorithm attempts to find an evasive point for up to 50 steps, and depending on the attack and oracle, some algorithms may have slower or more aggressive modification curves, which could explain the significant differences compared to Plot 2e.

7.6 Experiment 3: Attack Precomputation

The strategy described to generate the adversarial attacks requires repeated querying of a target autoencoder or predictor. This does not fit well with the speed of the CAN bus signals, therefore we test whether an attacker could compute a sequence of adversarial packets in advance and successfully inject it at a later time while avoiding detection. In practice, this experiment takes all those sequences that are *fully evasive*, i.e., exclusively composed of packets classified as normal traffic, and tries to find similar points in the flow of messages where they could equally evade the IDS. We also consider as candidate injection points every point in the traffic preceded by at least ten packets identical to the preamble found at the original attack location. We exclude from this test the FFNN model, as it performs classification independently of the order or position of messages, and CANdito as, given its superior resilience, there were not enough completely evasive sequences to carry out the test. Moreover, we provide this evaluation only for the ReCAN dataset, since the attacks on the CarHacking dataset are per-se injection attacks, and would anyway disrupt the normal flow of the network. The results are widely dependent on the specific CAN ID, with two clear clusters:

Cluster 1. IDs "1FB" and "104" have very slowly varying physval signals, that for the duration of the traffic gravitate around some common values rather than assuming any possible bit configuration (albeit all configurations are valid and no bit always remains constant). This behavior causes a relatively high number of possible reinjection points, with a peak cardinality of nearly 1800 points

found for a single sequence, and a high success rate. In general, sequences similar to the one in the view of Plot 2a transfer easily into spots in the traffic with a longer matching preamble, as we observe a 95% success rate with an average number of 38 identical preceding packets, obtaining several hundreds of potentially *fully evasive* sequences from 4 to 10 precomputed attacks, depending on the ID, attack, and oracle;

Cluster 2. The remaining 10 IDs do not bring the same degree of success as they provide way fewer injection points, with many preambles without a match in the whole traffic flow; once again, the few successful precomputed attacks require a preamble almost identical to the original, with over 37 matching messages on average. In general, this precomputed attack is feasible and reliable only within the trivial case of a preamble that is close to the original IDS input window, with more of the 90% of the successful reinjections differing only for a couple of packets. This makes just a few specific devices among the considered ECUs vulnerable to the approach under testing, however, it is not possible to ascertain the impact of the resulting risk with the current information about the function and semantic associated with each affected CAN ID.

7.7 Discussion on Defenses and Mitigation Strategies

While the focus of this work is on evaluating the vulnerabilities of machine learning-based intrusion detection systems (IDSs) in automotive networks, it is important to discuss potential mitigation strategies. The two main directions we envision are adversarial training and input preprocessing:

Adversarial training [11] involves incorporating adversarial examples during model training to improve robustness. This technique has shown promise in other domains such as computer vision, and could, in principle, be adapted to tabular and temporal domains like CAN traffic. However, applying adversarial training in the automotive setting presents unique challenges. In particular, introducing perturbed sequences during training may shift the model's decision boundary in a way that increases false positives. Given the safety-critical nature of automotive systems, an increase in false alarms may render an IDS practically unusable.

Input preprocessing techniques (e.g., smoothing, quantization, or feature selection) have been proposed in other fields [12] to reduce the impact of adversarial noise. Some of these approaches could be adapted to the CAN context, especially given its structured and discrete nature. However, their integration requires engineering to avoid excluding critical information from the detection process, which would allow the attacker to bypass the detection system by injecting attacks that do not affect the chosen features.

8 Conclusions

In this paper, we addressed the impact of adversarial attacks on state-of-the-art automotive IDSs. We conducted a thorough evaluation of known adversarial evasion attacks, adapted to the automotive domain, against payload-based IDSs using real CAN traffic over two public datasets. We designed and implemented customized variants of the popular BIM and DeepFool perturbation algorithms, and tested six different detection architectures from the state of the art in white-, grey-, and black-box scenarios. Our results show that evasion is achievable, although not always consistently, especially in the white-box scenario, across both datasets. In the grey- and black-box scenarios, the performance degradation of the intrusion detection systems is notably reduced. DeepFool proves to be the most effective evasion algorithm, generating numerous evasion points on both datasets and achieving up to an 85% drop in TPR. Exploring the characteristics of evasion samples, we observe that in multiple instances, either part of the attack sequence fails to evade detection or the perturbation effects render the payload similar to a non-tampered one. We evaluate the feasibility of precomputing the evasive sequence and injecting it, demonstrating that this is often possible.

The most significant limitation of our study stems from the lack of availability of a real test vehicle, where the injected packets may influence the vehicle's and bus's behavior—a characteristic that cannot be evaluated with previously collected traffic logs. Future work will aim to evaluate how well perturbed attack sequences preserve the original attack intent, providing a more comprehensive understanding of evasion algorithms—not only in terms of bypassing detection but also in achieving the attacker's intended impact. Additionally, further research should investigate the applicability and effectiveness of alternative adversarial approaches, such as score-based methods, in tabular and temporal domains like those found in automotive systems.

Acknowledgments

This work was partially supported by projects SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU and MICS (Made in Italy – Circular and Sustainable) from Next-Generation EU. CUP MICS D43C22003120001.

References

[1] Malik Avatefipour. [n. d.]. State-of-the-Art Survey on In-Vehicle Network Communication "CAN-Bus" Security and Vulnerabilities. ([n. d.]).

[2] Paolo Cerracchio, Stefano Longari, Michele Carminati, Stefano Zanero, et al. 2024. Investigating the Impact of Evasion Attacks Against Automotive Intrusion Detection Systems. In *Symposium on Vehicles Security and Privacy (VehicleSec) 2024*. N–A.

[3] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial Attacks and Defences: A Survey. arXiv:1810.00069 [cs.LG]

[4] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, and Chiew Tong Lau. 2018. Autoencoder-based network anomaly detection. In *2018 Wireless Telecommunications Symposium (WTS)*. 1–5. https://doi.org/10.1109/WTS.2018.8363930

[5] Valliappa Chockalingam, Ian Larson, Daniel Lin, and Spencer Nofzinger. 2016. Detecting attacks on the CAN protocol with machine learning. *Annu EECS* 558, 7 (2016).

[6] Cia. [n. d.]. *CAN data link layers in some detail.* https://www.can-cia.org/can-knowledge/can/can-data-link-layers/

[7] Alvise de Faveri Tron, Stefano Longari, Michele Carminati, Mario Polino, and Stefano Zanero. 2022. CANflict: Exploiting Peripheral Conflicts for Data-Link Layer Attacks on Automotive Networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (Eds.). ACM, 711–723. https://doi.org/10.1145/3548606.3560618

[8] Dorothy E. Denning. 1987. An Intrusion-Detection Model. *IEEE Trans. Software Eng.* 13, 2 (1987), 222–232. https://doi.org/10.1109/TSE.1987.232894

[9] Eric Evenchick. [n. d.]. CANtact: Open Source Car Tool. https://cantact.io/. Accessed: 2025-03-21.

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6572

[12] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. 2017. Countering Adversarial Images using Input Transformations. *CoRR* abs/1711.00117 (2017). arXiv:1711.00117 http://arxiv.org/abs/1711.00117

[13] Markus Hanselmann, Thilo Strauss, Katharina Dormann, and Holger Ulmer. 2020. CANet: An unsupervised intrusion detection system for high dimensional CAN bus data. *Ieee Access* 8 (2020), 58194–58205.

[14] Md Delwar Hossain, Hiroyuki Inoue, Hideya Ochiai, Doudou Fall, and Youki Kadobayashi. 2020. An effective in-vehicle CAN bus intrusion detection system using CNN deep learning approach. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 1–6.

[15] Texas Instruments. 2002. Introductionto the ControllerAreaNetwork(CAN). (2002).

[16] Min-Joo Kang and Je-Won Kang. 2016. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one* 11, 6 (2016), e0155781.

[17] Zadid Khan, Mashrur Chowdhury, Mhafuzul Islam, Chin-Ya Huang, and Mizanur Rahman. 2019. Long short-term memory neural networks for false information attack detection in software-defined in-vehicle network. *arXiv preprint arXiv:1906.10203* (2019).

[18] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. 2018. Adversarial examples in the physical world. In *Artificial intelligence safety and security*. Chapman and Hall/CRC, 99–112. From 2016 preprint.

[19] Brooke Lampe and Weizhi Meng. 2023. A survey of deep learning-based intrusion detection in automotive applications. *Expert Systems with Applications* (2023), 119771.

[20] Yi Li, Jing Lin, and Kaiqi Xiong. 2021. An adversarial attack defending system for securing in-vehicle networks. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 1–6.

[21] Stefano Longari, Alessandro Nichelini, Carlo Alberto Pozzoli, Michele Carminati, and Stefano Zanero. 2022. CANdito: Improving Payload-based Detection of Attacks on Controller Area Networks. *arXiv preprint arXiv:2208.06628* (2022).

[22] Stefano Longari, Francesco Noseda, Michele Carminati, and Stefano Zanero. 2023. Evaluating the Robustness of Automotive Intrusion Detection Systems Against Evasion Attacks. In *International Symposium on Cyber Security, Cryptology, and Machine Learning*. Springer, 337–352.

[23] Stefano Longari, Daniel Humberto Nova Valcarcel, Mattia Zago, Michele Carminati, and Stefano Zanero. 2020. CANnolo: An anomaly detection system based on LSTM autoencoders for controller area network. *IEEE Transactions on Network and Service Management* 18, 2 (2020), 1913–1924.

[24] Mirco Marchetti and Dario Stabili. 2018. READ: Reverse Engineering of Automotive Data Frames. *IEEE Transactions on Information Forensics and Security* (09 2018). https://doi.org/10.1109/TIFS.2018.2870826

[25] Charlie Miller and Chris Valasek. 2015. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* (2015).

[26] Charlie Miller and Chris Valasek. 2016. CAN Message Injection. https://illmatics.com/can%20message%20injection.pdf [Online, accessed 1-Oct-2022].

[27] Michael R Moore, Robert A Bridges, Frank L Combs, Michael S Starr, and Stacy J Prowell. 2017. Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks: a data-driven approach to in-vehicle intrusion detection. In *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*. 1–4.

[28] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2574–2582.

[29] Alessandro Nichelini, Carlo Alberto Pozzoli, Stefano Longari, Michele Carminati, and Stefano Zanero. 2023. CANova: A hybrid intrusion detection framework based on automatic signal classification for CAN. *Computers and Security* 128 (2023). https://doi.org/10.1016/j.cose.2023.103166

[30] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 506–519.

[31] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*. IEEE, 582–597.

[32] Sampath Rajapaksha, Harsha Kalutarage, M Omar Al-Kadri, Andrei Petrovski, Garikayi Madzudzo, and Madeline Cheah. 2023. Ai-based intrusion detection systems for in-vehicle networks: A survey. *Comput. Surveys* 55, 11 (2023).

[33] Rafi Ud Daula Refat, Abdulrahman Abu Elkhail, Azeem Hafeez, and Hafiz Malik. 2022. Detecting CAN bus intrusion by applying machine learning method to graph based features. In *Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys) Volume 3*. Springer, 730–748.

[34] Kudzai Sauka, Gun-Yoo Shin, Dong-Wook Kim, and Myung-Mook Han. 2022. Adversarial robust and explainable network intrusion detection systems based on deep learning. *Applied Sciences* 12, 13 (2022), 6451.

[35] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. 2018. GIDS: GAN based intrusion detection system for in-vehicle network. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 1–6.

[36] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. 2020. Adversarial Machine Learning-Industry Perspectives. In *2020 IEEE Security and Privacy Workshops (SPW)*. 69–75. https://doi.org/10.1109/SPW50608.2020.00028

[37] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. 2020. In-vehicle network intrusion detection using deep convolutional neural network. *Vehicular Communications* 21 (2020), 100198.

[38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. https://doi.org/10.48550/ARXIV.1312.6199

[39] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1312.6199

[40] Vinayak Tanksale. 2020. Anomaly Detection for Controller Area Networks Using Long Short-Term Memory. *IEEE Open Journal of Intelligent Transportation Systems* 1 (2020), 253–265. https://doi.org/10.1109/OJITS.2020.3043066

[41] Adrian Taylor, Nathalie Japkowicz, and Sylvain Leblanc. 2015. Frequency-based anomaly detection for the automotive CAN bus. In *2015 World Congress on Industrial Control Systems Security (WCICSS)*. 45–49. https://doi.org/10.1109/WCICSS.2015.7420322

[42] Adrian Taylor, Sylvain Leblanc, and Nathalie Japkowicz. 2016. Anomaly detection in automobile control network data with long short-term memory networks. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 130–139.

[43] Andrew Tomlinson, Jeremy Bryans, Siraj Ahmed Shaikh, and Harsha Kumara Kalutarage. 2018. Detection of automotive CAN cyber-attacks by identifying packet timing anomalies in time windows. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 231–238.

[44] Miki E. Verma, Michael D. Iannacone, Robert A. Bridges, Samuel C. Hollifield, Pablo Moriano, Bill Kay, and Frank L. Combs. 2022. Addressing the Lack of Comparability & Testing in CAN Intrusion Detection Research: A Comprehensive Guide to CAN IDS Data & Introduction of the ROAD Dataset. arXiv:2012.14600 [cs.CR]

[45] Eric Wong, Leslie Rice, and J. Zico Kolter. 2020. Fast is better than free: Revisiting adversarial training. arXiv:2001.03994 [cs.LG]

[46] Kaichao You, Mingsheng Long, Jianmin Wang, and Michael I Jordan. 2019. How does learning rate decay help modern neural networks? *arXiv preprint arXiv:1908.01878* (2019).

[47] Mattia Zago, Stefano Longari, Andrea Tricarico, Michele Carminati, Manuel Gil Pérez, Gregorio Martìnez Pérez, and Stefano Zanero. 2020. ReCAN–dataset for reverse engineering of controller area networks. *Data in brief* 29 (2020), 105149.

Received 15 September 2024; revised 1 February 2025; accepted 18 May 2025